OIPE
AUG 27 2004
JC95
PATENT & TRADEMARK OFFICE

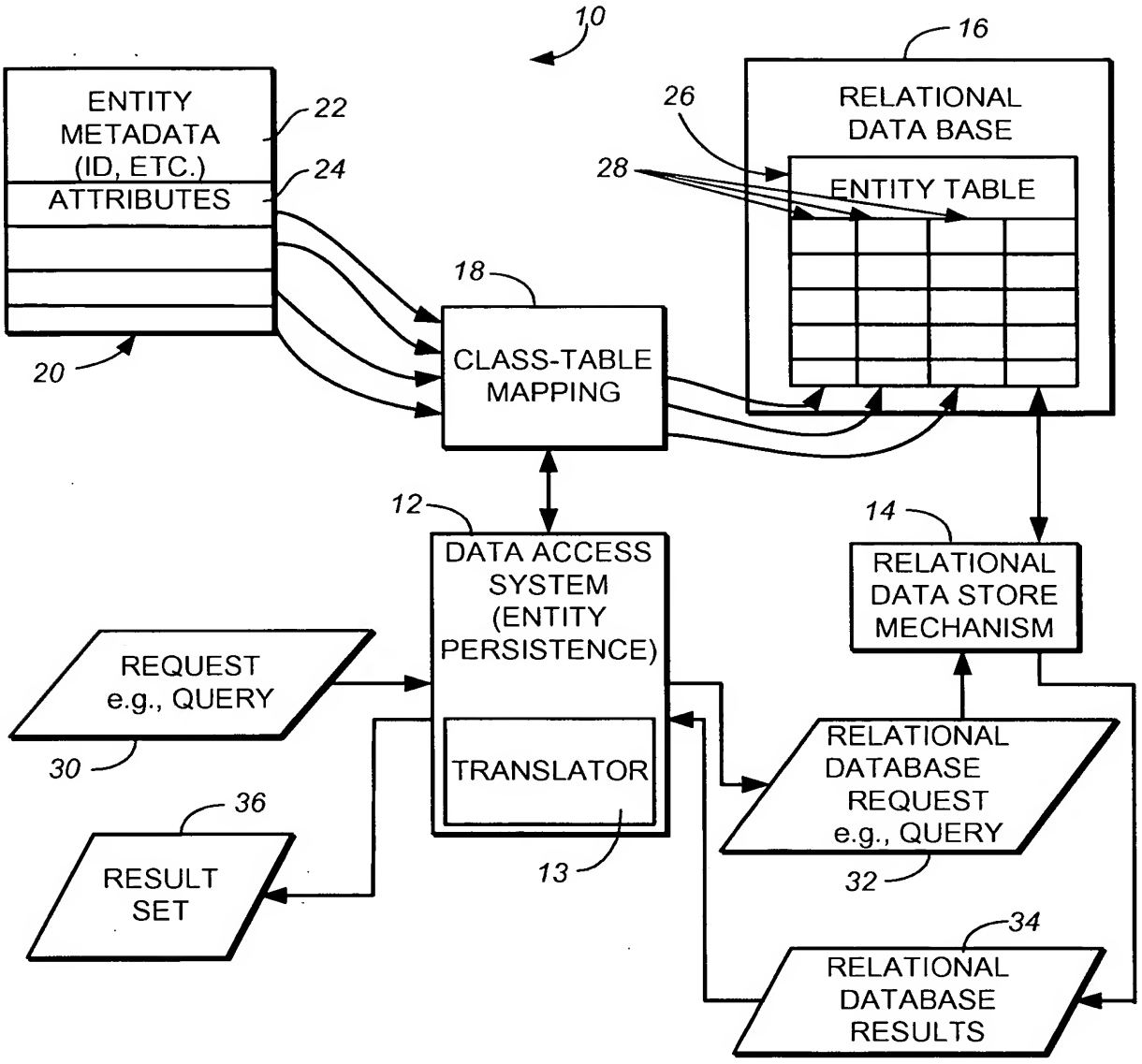


FIG._1

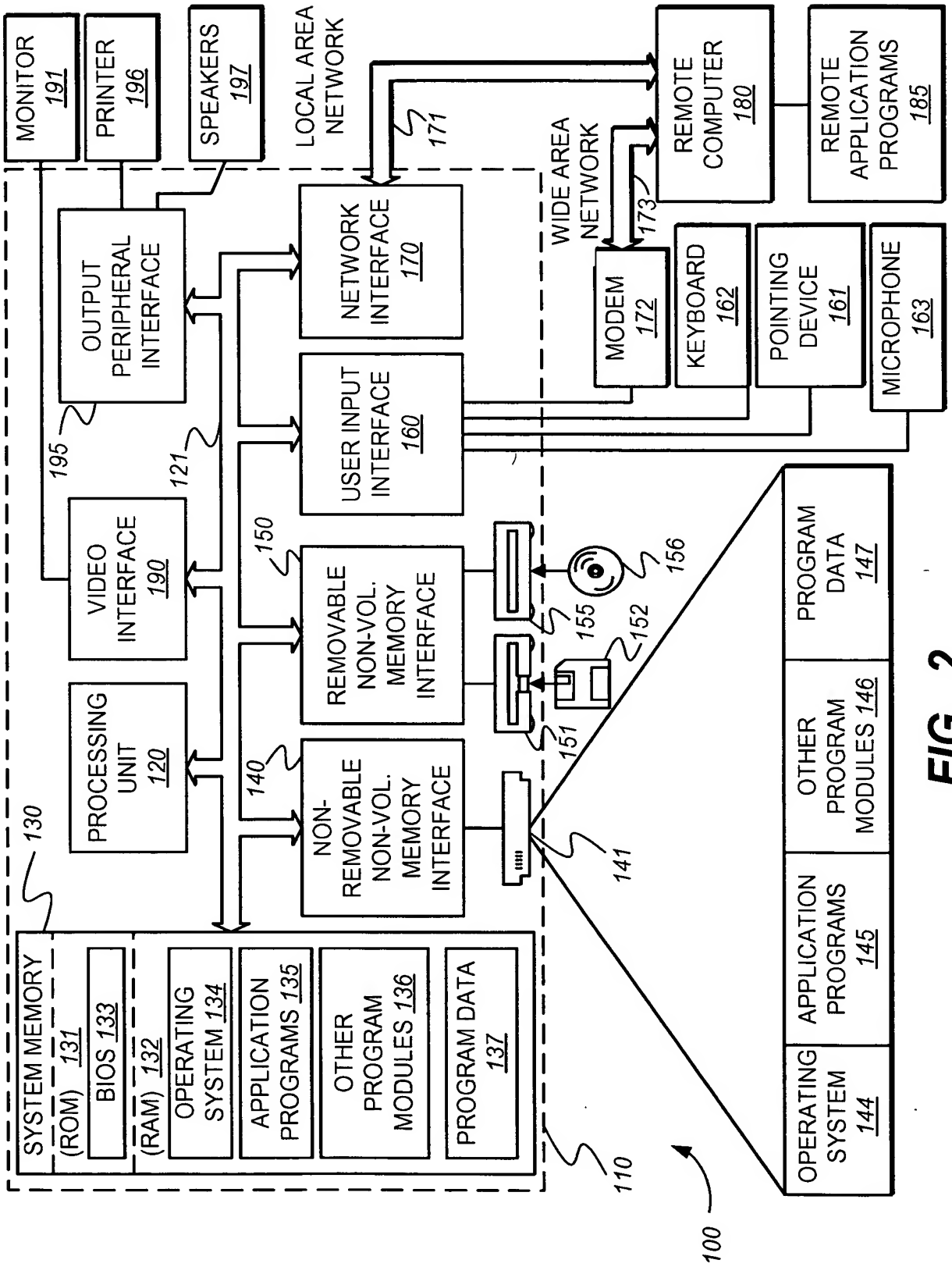

MONITOR 191
PRINTER 196
SPEAKERS 197
LOCAL AREA NETWORK
171
WIDE AREA NETWORK
REMOTE COMPUTER 180
REMOTE APPLICATION PROGRAMS 185
OUTPUT PERIPHERAL INTERFACE
NETWORK INTERFACE 170
MODEM 172
KEYBOARD 162
POINTING DEVICE 161
MICROPHONE 163
USER INPUT INTERFACE 160
195
121
VIDEO INTERFACE 190
150
REMOVABLE NON-VOL. MEMORY INTERFACE
155
156
152
151
PROCESSING UNIT 120
140
NON-REMOVABLE NON-VOL. MEMORY INTERFACE
130
141
SYSTEM MEMORY
(ROM) 131
BIOS 133
(RAM) 132
OPERATING SYSTEM 134
APPLICATION PROGRAMS 135
OTHER PROGRAM MODULES 136
PROGRAM DATA 137
110
100
PROGRAM DATA 147
OTHER PROGRAM MODULES 146
APPLICATION PROGRAMS 145
OPERATING SYSTEM 144

FIG._2

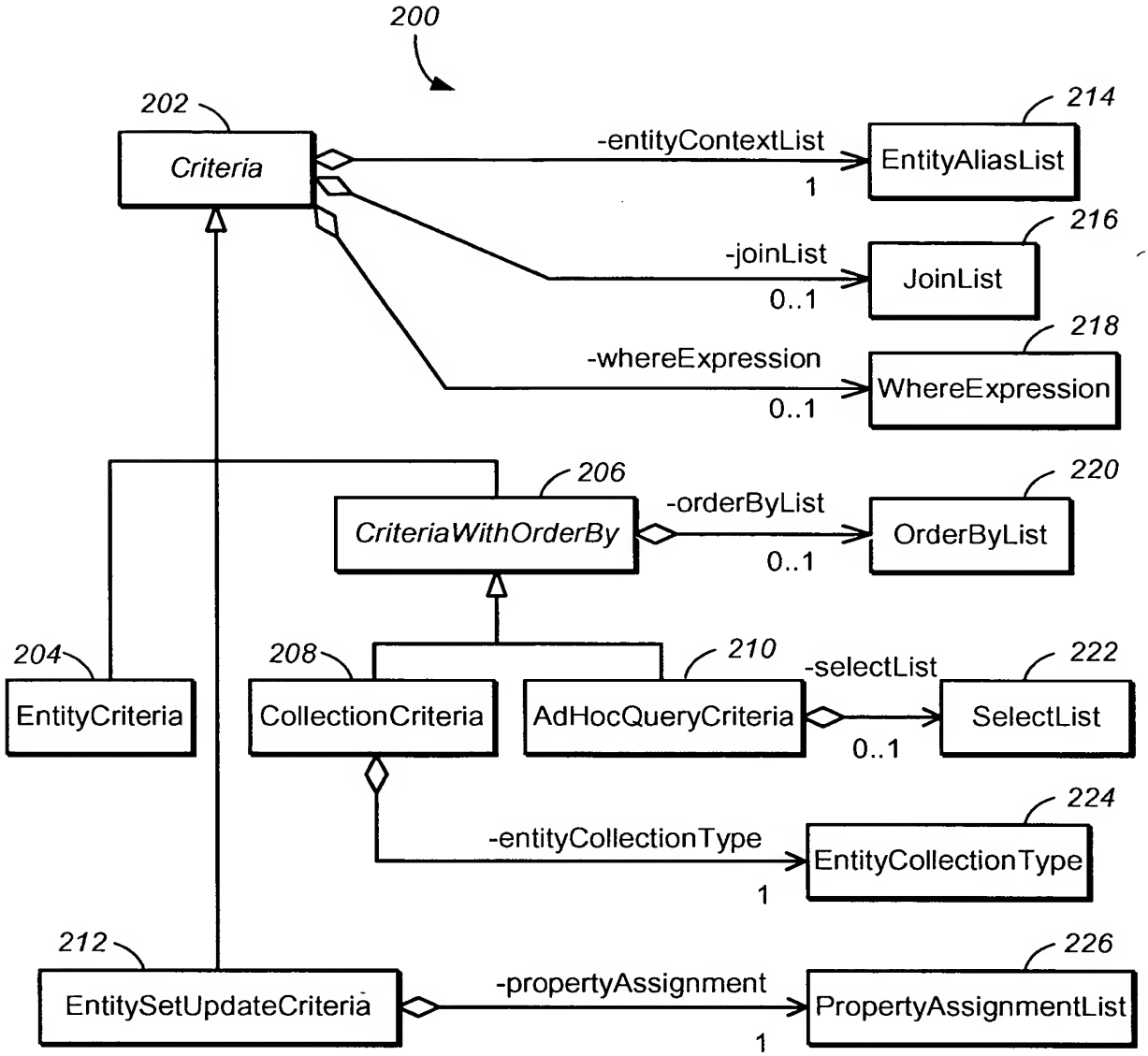

200
202
Criteria
214
EntityAliasList
-entityContextList
1
216
JoinList
-joinList
0..1
218
WhereExpression
-whereExpression
0..1
206
CriteriaWithOrderBy
220
OrderByList
-orderByList
0..1
204
EntityCriteria
208
CollectionCriteria
210
AdHocQueryCriteria
-selectList
0..1
222
SelectList
224
EntityCollectionType
-entityCollectionType
1
212
EntitySetUpdateCriteria
226
PropertyAssignmentList
-propertyAssignment
1

FIG._3

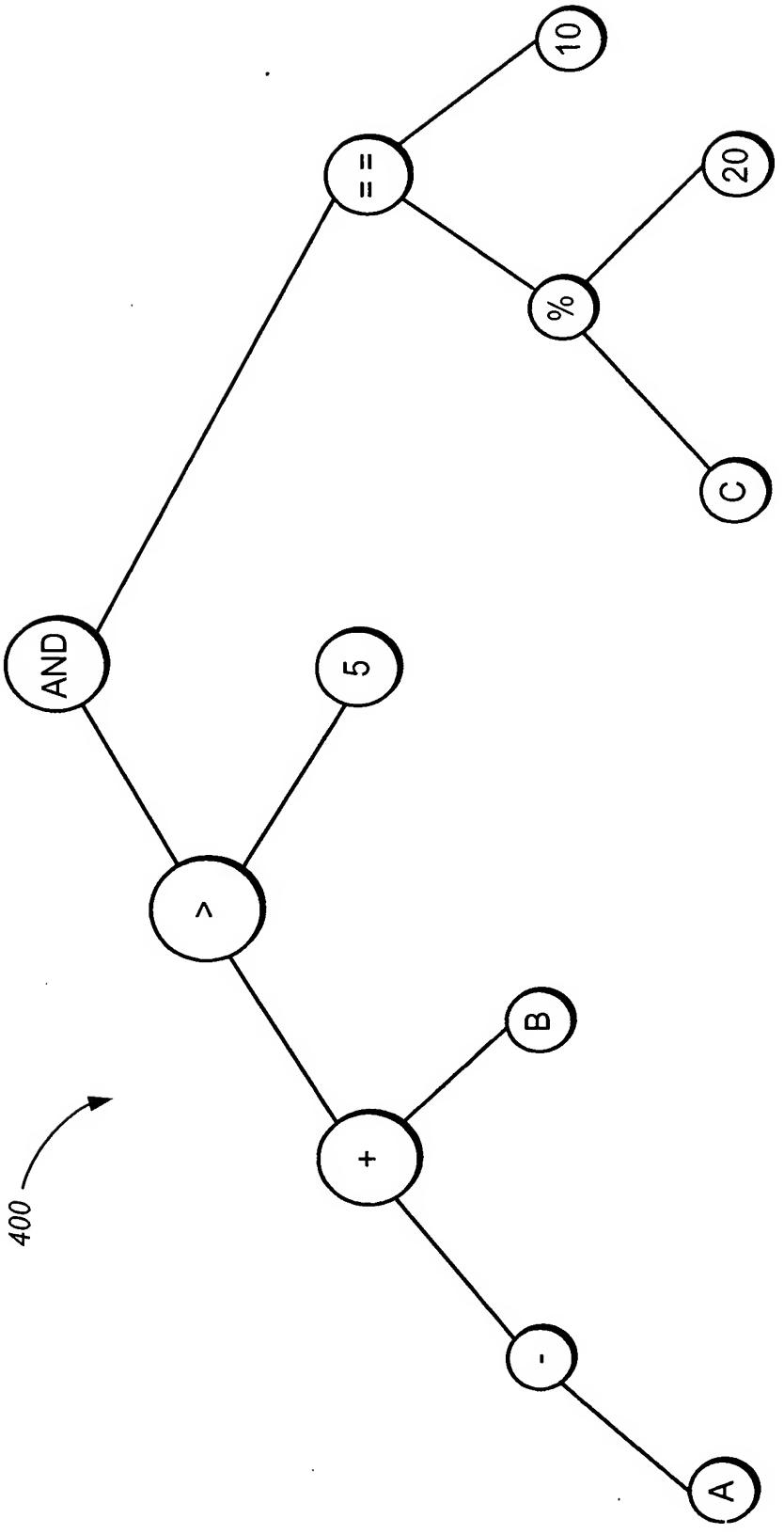


FIG._4A

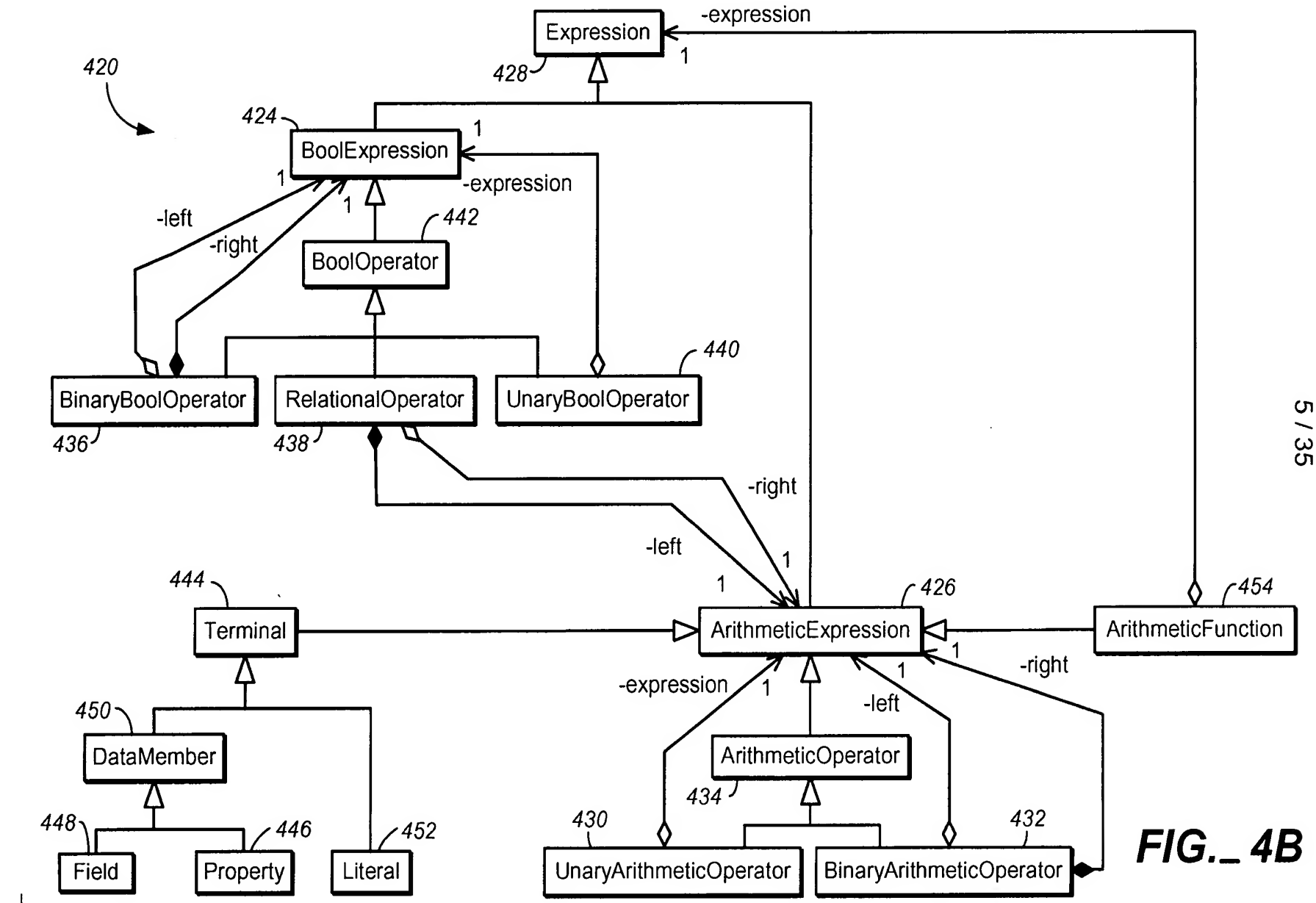

420
Expression
-expression
1
428
424
BoolExpression
1
1
-expression
1
-left
-right
442
BoolOperator
440
BinaryBoolOperator
RelationalOperator
UnaryBoolOperator
436
438
-right
-left
1
1
444
426
454
Terminal
ArithmeticExpression
ArithmeticFunction
1
1
-right
-expression
1
-left
450
DataMember
ArithmeticOperator
434
448
446
452
430
432
Field
Property
Literal
UnaryArithmeticOperator
BinaryArithmeticOperator

FIG._4B

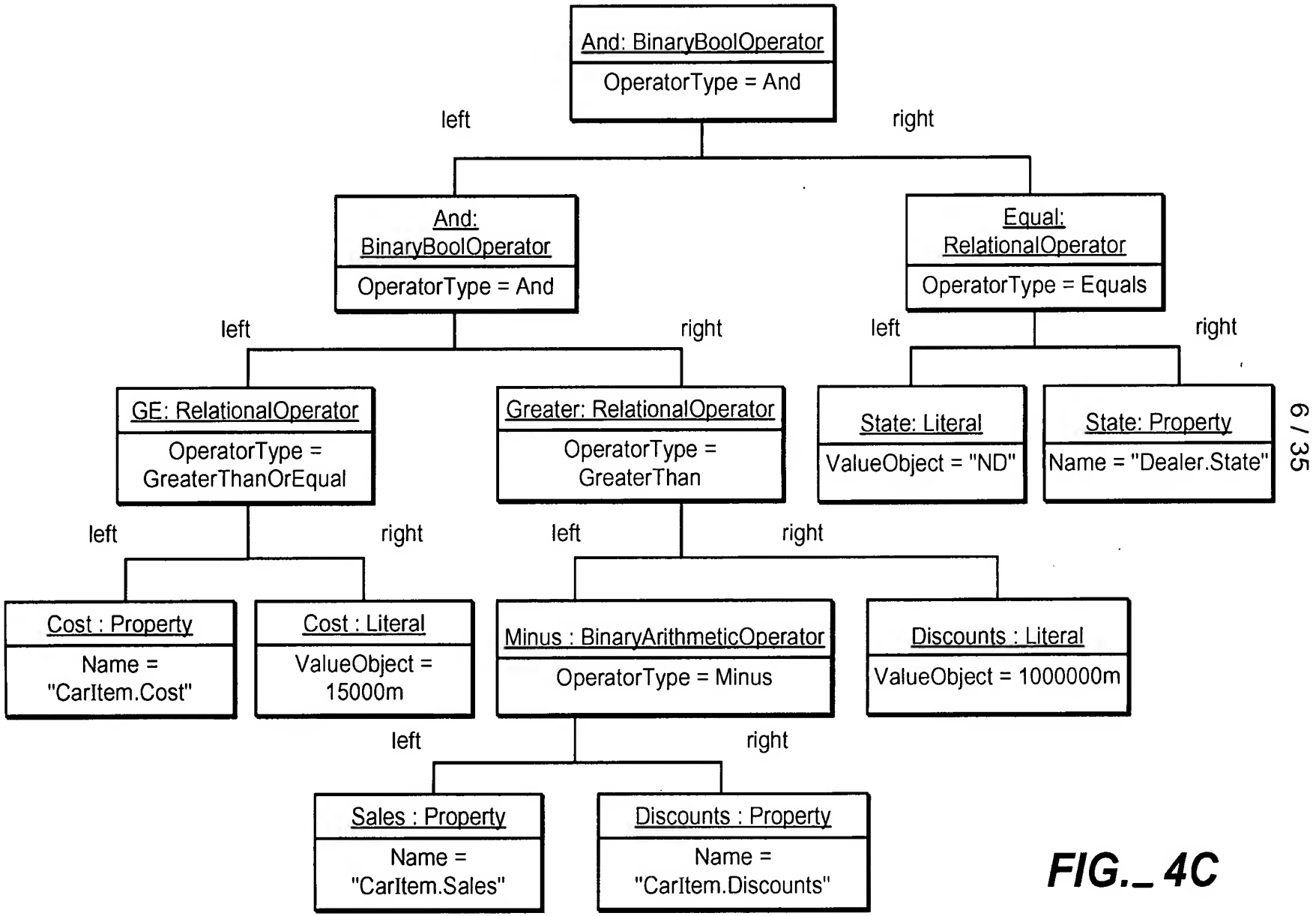

And: BinaryBoolOperator
OperatorType = And
left
right
And: BinaryBoolOperator
OperatorType = And
Equal: RelationalOperator
OperatorType = Equals
left
right
left
right
GE: RelationalOperator
OperatorType = GreaterThanOrEqual
Greater: RelationalOperator
OperatorType = GreaterThan
State: Literal
ValueObject = "ND"
State: Property
Name = "Dealer.State"
left
right
left
right
Cost : Property
Name = "CarItem.Cost"
Cost : Literal
ValueObject = 15000m
Minus : BinaryArithmeticOperator
OperatorType = Minus
Discounts : Literal
ValueObject = 1000000m
left
right
Sales : Property
Name = "CarItem.Sales"
Discounts : Property
Name = "CarItem.Discounts"

FIG._4C

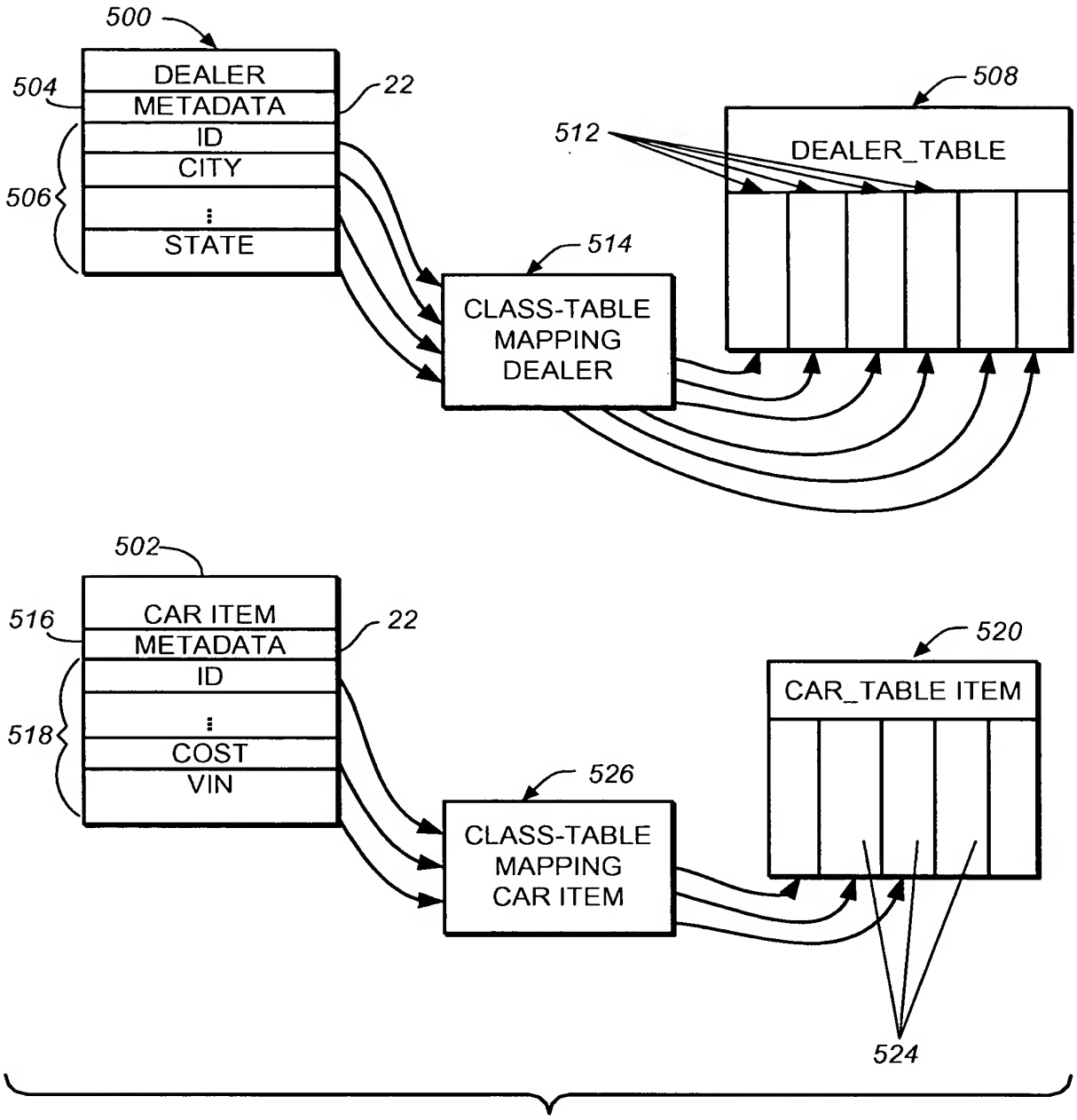

500
DEALER
METADATA
ID
CITY
STATE
504
506
22
508
512
DEALER_TABLE
514
CLASS-TABLE
MAPPING
DEALER
502
CAR ITEM
METADATA
ID
COST
VIN
516
518
22
520
CAR_TABLE ITEM
526
CLASS-TABLE
MAPPING
CAR ITEM
524

FIG._5

```
class CarItem {     // these properties are mapped to a database table
        public string ID
        public string Vin;
        public string Color;
        public decimal Cost;
        ... many others omitted ...
}
class Dealer {      // these properties are mapped to a different database table
        public string ID;
        public string City;
        public string State;
        ... many others omitted ...
}

AdHocQueryCriteria  adHocCriteria = Criteria.AdHocQueryCriteria(
        Criteria.EntityAliases(     // describes the objects involved in the query
                Criteria.EntityAlias(itemParentKey, typeof(CarItem))
                Criteria.EntityAlias(dealerParentKey, typeof(Dealer)) ),
        Criteria.JoinList(
                        /* entity to entity join */
                Criteria.InnerJoin("CarItem", "Dealer",
                (Property)"CarItem.DealerID" == (Property)"Dealer.ID")),
        Criteria.Select(            // the specific properties to retrieve
                (Property)"CarItem.ID",  // references the field in the above class
                (Property)"CarItem.Vin",
                (Property)"CarItem.Cost",
                (Property)"Dealer.ID",
                (Property)"Dealer.City",
                (Property)"Dealer.State"),
        Criteria.Where(
                (Property)"CarItem.Make" == "Geo" &&
                (Property)"CarItem.Model" == "Prism" &&
                (Property)"Dealer.State" == "ND")
        Criteria.OrderBy((Property)"Dealer.Cost"));
```

FIG._6

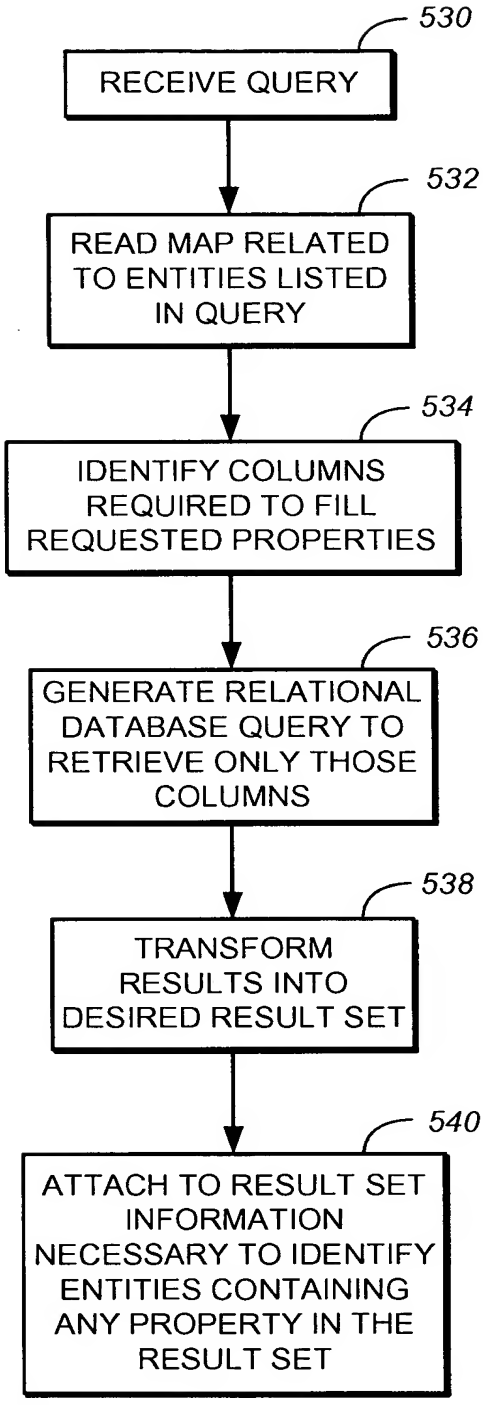

530
RECEIVE QUERY
532
READ MAP RELATED TO ENTITIES LISTED IN QUERY
534
IDENTIFY COLUMNS REQUIRED TO FILL REQUESTED PROPERTIES
536
GENERATE RELATIONAL DATABASE QUERY TO RETRIEVE ONLY THOSE COLUMNS
538
TRANSFORM RESULTS INTO DESIRED RESULT SET
540
ATTACH TO RESULT SET INFORMATION NECESSARY TO IDENTIFY ENTITIES CONTAINING ANY PROPERTY IN THE RESULT SET

FIG._7

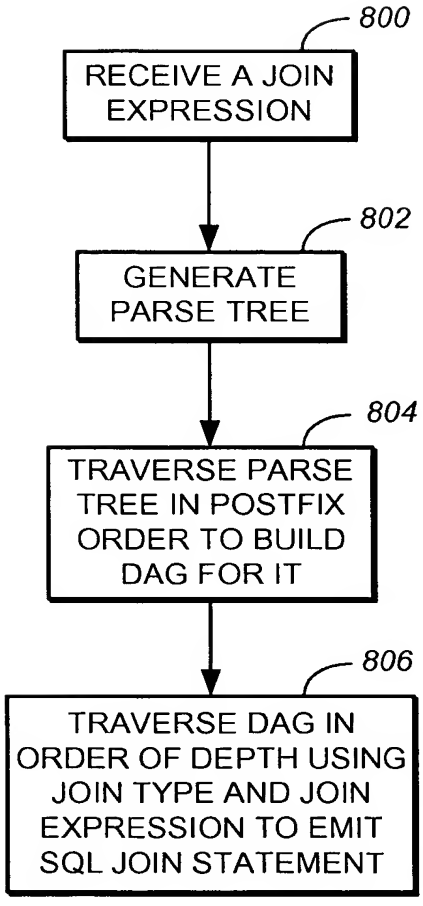

800
RECEIVE A JOIN EXPRESSION
802
GENERATE PARSE TREE
804
TRAVERSE PARSE TREE IN POSTFIX ORDER TO BUILD DAG FOR IT
806
TRAVERSE DAG IN ORDER OF DEPTH USING JOIN TYPE AND JOIN EXPRESSION TO EMIT SQL JOIN STATEMENT

FIG._8

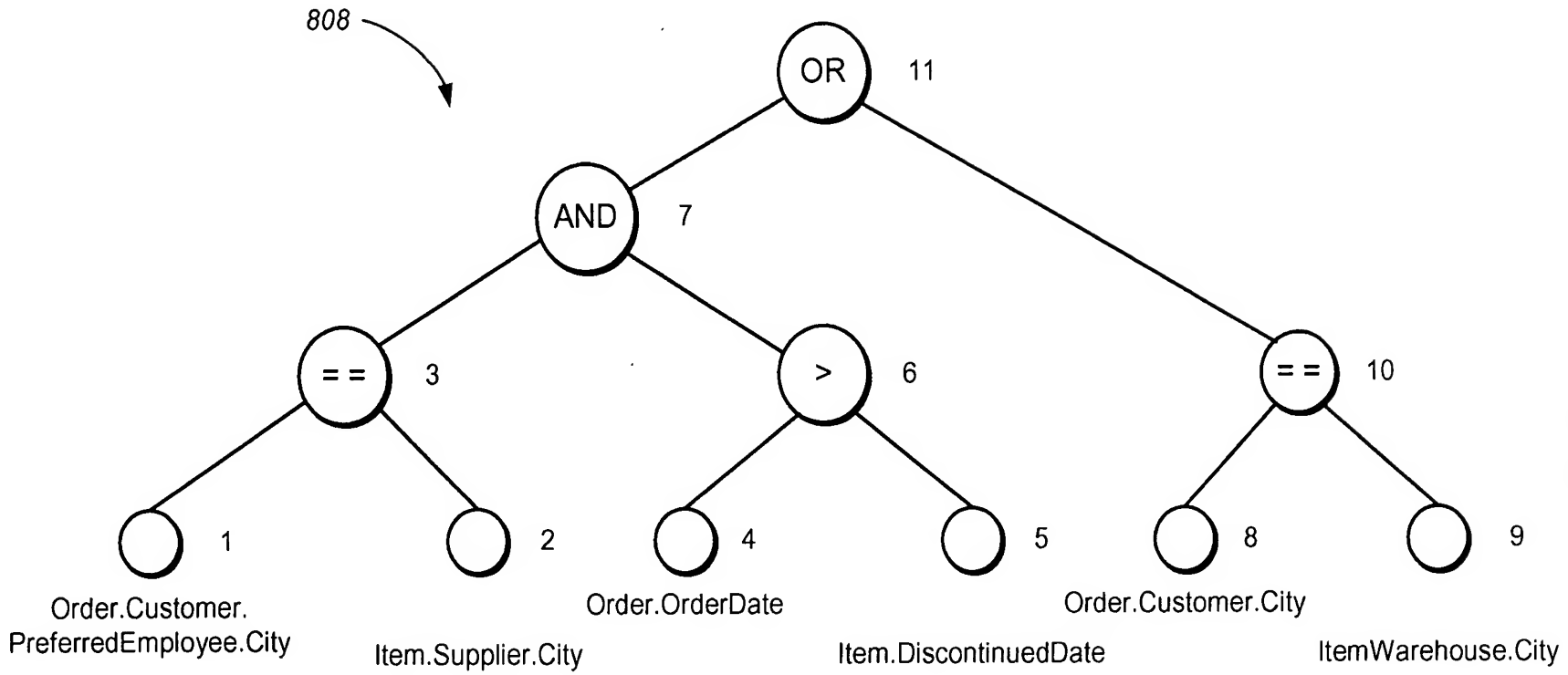
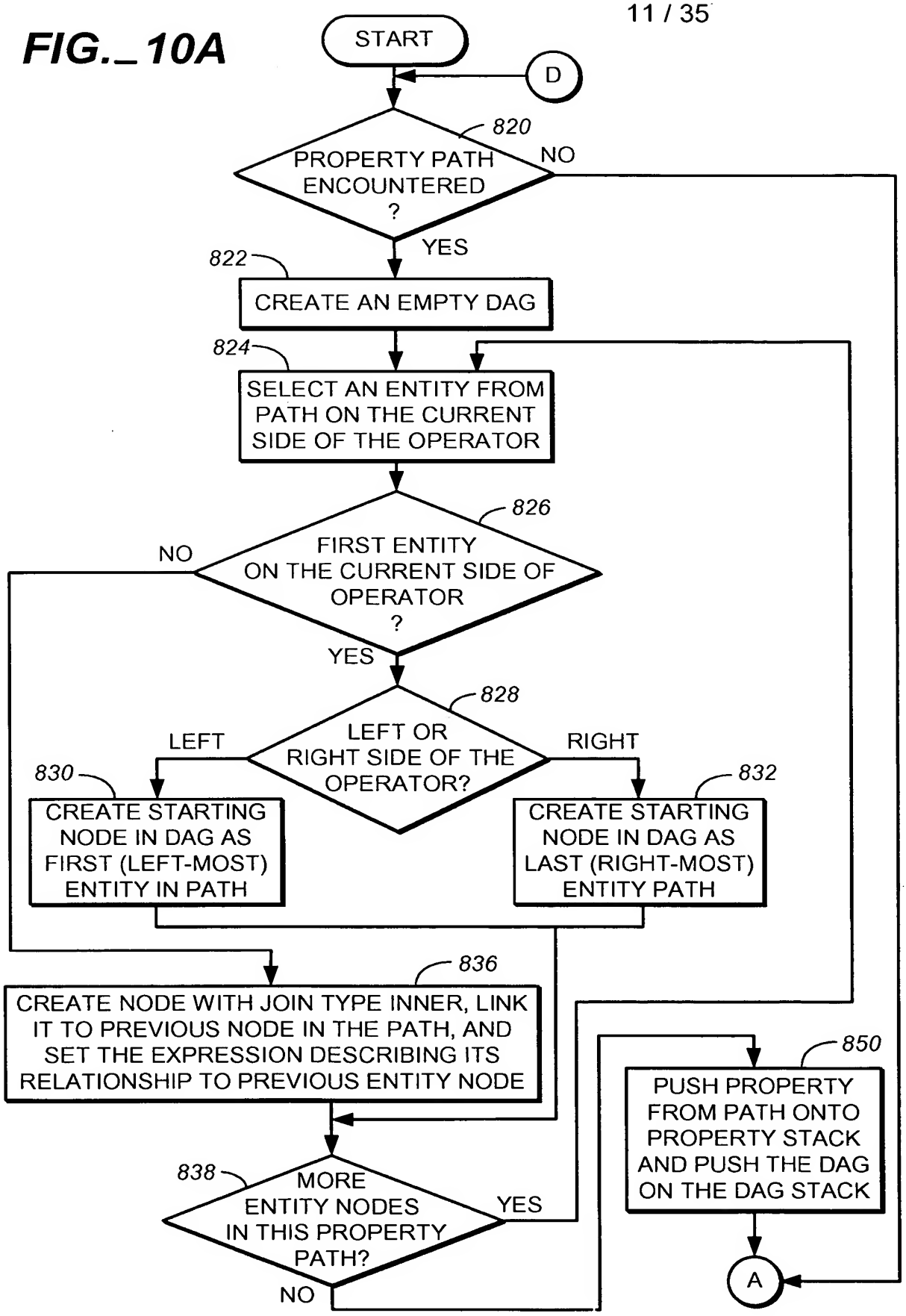

808
OR
11
AND
7
==
3
>
6
==
10
1
2
4
5
8
9
Order.Customer.
PreferredEmployee.City
Item.Supplier.City
Order.OrderDate
Item.DiscontinuedDate
Order.Customer.City
ItemWarehouse.City

FIG._ 9

FIG._10A


START
D
820
PROPERTY PATH ENCOUNTERED ?
NO
YES
822
CREATE AN EMPTY DAG
824
SELECT AN ENTITY FROM PATH ON THE CURRENT SIDE OF THE OPERATOR
826
FIRST ENTITY ON THE CURRENT SIDE OF OPERATOR ?
NO
YES
828
LEFT OR RIGHT SIDE OF THE OPERATOR?
LEFT
RIGHT
830
CREATE STARTING NODE IN DAG AS FIRST (LEFT-MOST) ENTITY IN PATH
832
CREATE STARTING NODE IN DAG AS LAST (RIGHT-MOST) ENTITY PATH
836
CREATE NODE WITH JOIN TYPE INNER, LINK IT TO PREVIOUS NODE IN THE PATH, AND SET THE EXPRESSION DESCRIBING ITS RELATIONSHIP TO PREVIOUS ENTITY NODE
838
MORE ENTITY NODES IN THIS PROPERTY PATH?
YES
NO
850
PUSH PROPERTY FROM PATH ONTO PROPERTY STACK AND PUSH THE DAG ON THE DAG STACK
A

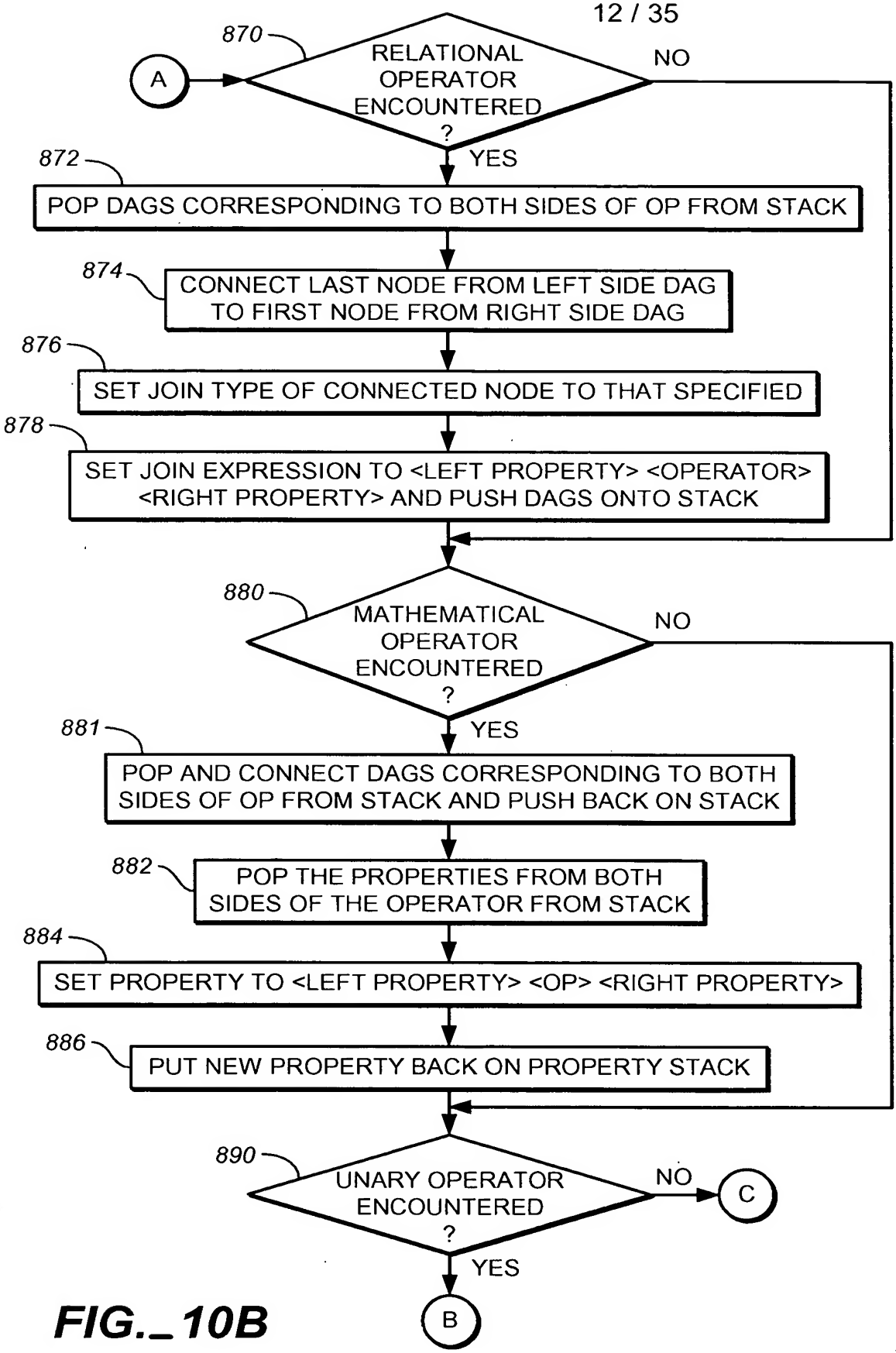
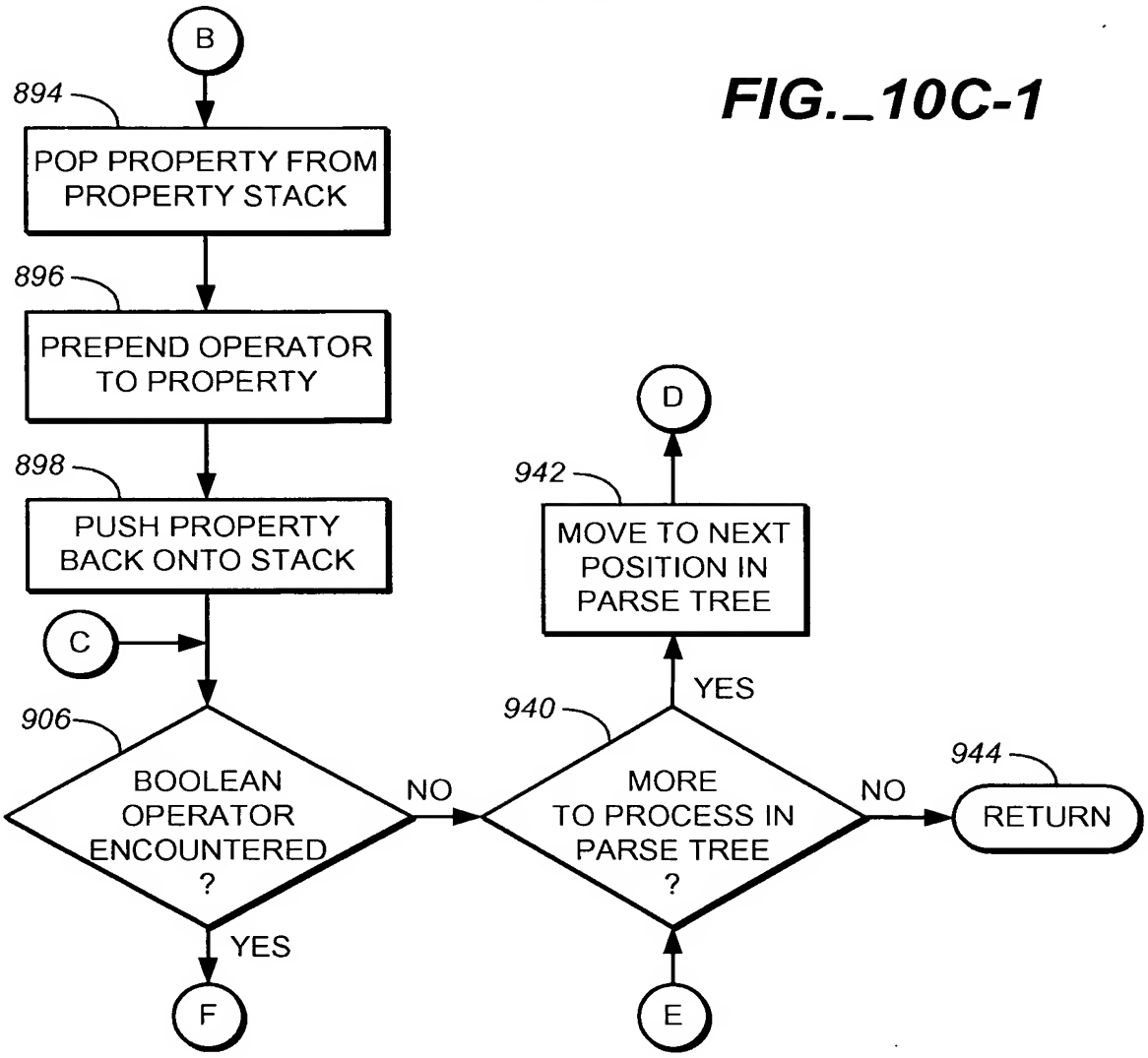

870
RELATIONAL OPERATOR ENCOUNTERED ?
NO
A
YES
872
POP DAGS CORRESPONDING TO BOTH SIDES OF OP FROM STACK
874
CONNECT LAST NODE FROM LEFT SIDE DAG TO FIRST NODE FROM RIGHT SIDE DAG
876
SET JOIN TYPE OF CONNECTED NODE TO THAT SPECIFIED
878
SET JOIN EXPRESSION TO <LEFT PROPERTY> <OPERATOR> <RIGHT PROPERTY> AND PUSH DAGS ONTO STACK
880
MATHEMATICAL OPERATOR ENCOUNTERED ?
NO
YES
881
POP AND CONNECT DAGS CORRESPONDING TO BOTH SIDES OF OP FROM STACK AND PUSH BACK ON STACK
882
POP THE PROPERTIES FROM BOTH SIDES OF THE OPERATOR FROM STACK
884
SET PROPERTY TO <LEFT PROPERTY> <OP> <RIGHT PROPERTY>
886
PUT NEW PROPERTY BACK ON PROPERTY STACK
890
UNARY OPERATOR ENCOUNTERED ?
NO
C
YES
B

FIG._10B

FIG._10C-1


B
894
POP PROPERTY FROM PROPERTY STACK
896
PREPEND OPERATOR TO PROPERTY
898
PUSH PROPERTY BACK ONTO STACK
C
906
BOOLEAN OPERATOR ENCOUNTERED ?
NO
YES
F
D
942
MOVE TO NEXT POSITION IN PARSE TREE
YES
940
MORE TO PROCESS IN PARSE TREE ?
NO
E
944
RETURN

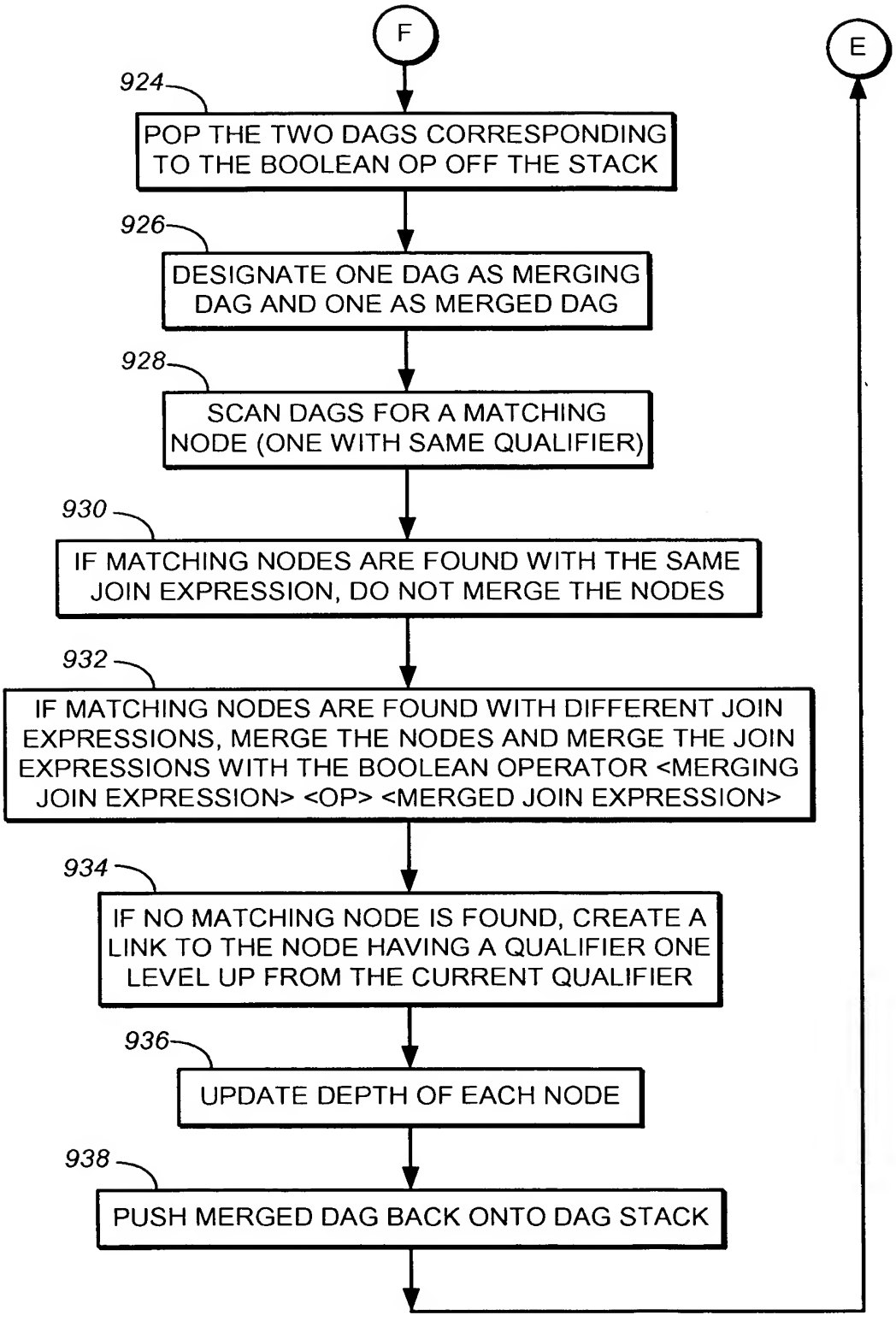

F
E
924
POP THE TWO DAGS CORRESPONDING TO THE BOOLEAN OP OFF THE STACK
926
DESIGNATE ONE DAG AS MERGING DAG AND ONE AS MERGED DAG
928
SCAN DAGS FOR A MATCHING NODE (ONE WITH SAME QUALIFIER)
930
IF MATCHING NODES ARE FOUND WITH THE SAME JOIN EXPRESSION, DO NOT MERGE THE NODES
932
IF MATCHING NODES ARE FOUND WITH DIFFERENT JOIN EXPRESSIONS, MERGE THE NODES AND MERGE THE JOIN EXPRESSIONS WITH THE BOOLEAN OPERATOR <MERGING JOIN EXPRESSION> <OP> <MERGED JOIN EXPRESSION>
934
IF NO MATCHING NODE IS FOUND, CREATE A LINK TO THE NODE HAVING A QUALIFIER ONE LEVEL UP FROM THE CURRENT QUALIFIER
936
UPDATE DEPTH OF EACH NODE
938
PUSH MERGED DAG BACK ONTO DAG STACK

FIG._10C-2

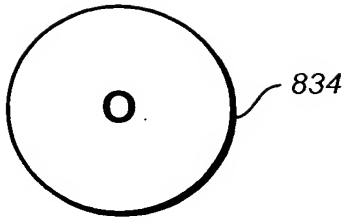


FIG._11A

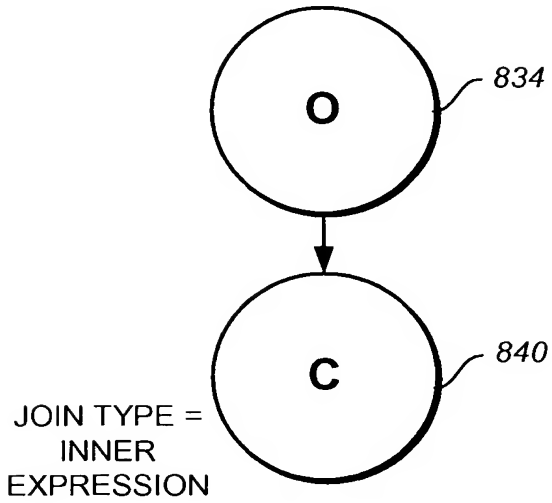


FIG._11B

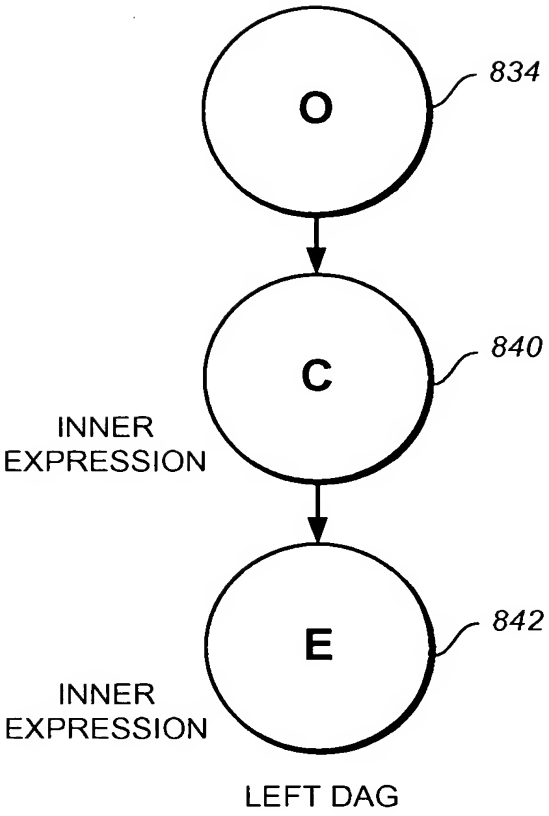


FIG._11C

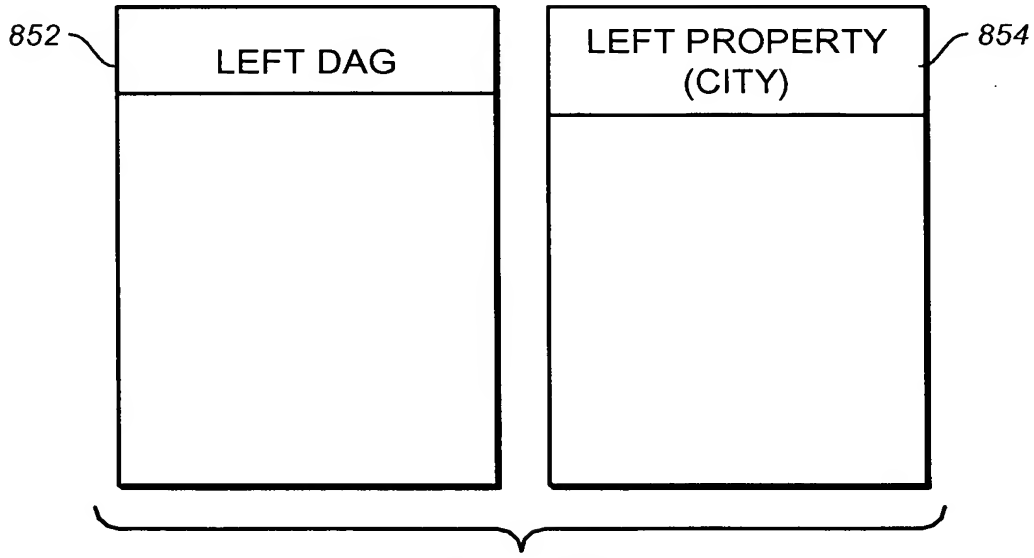


FIG._11D

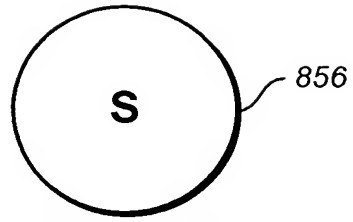


FIG._11E

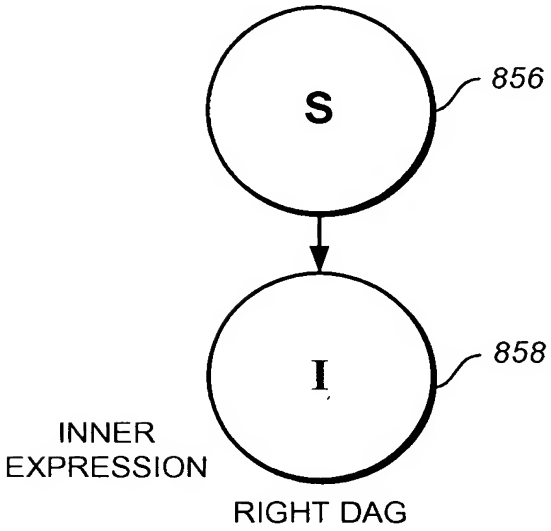


FIG._11F

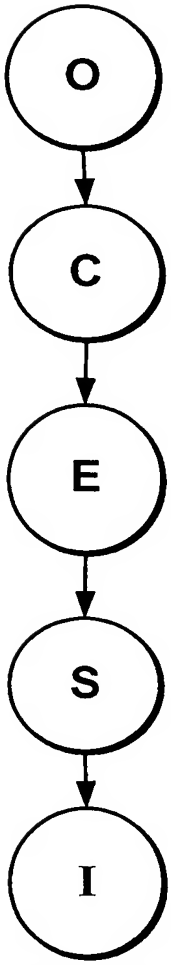


FIG._12

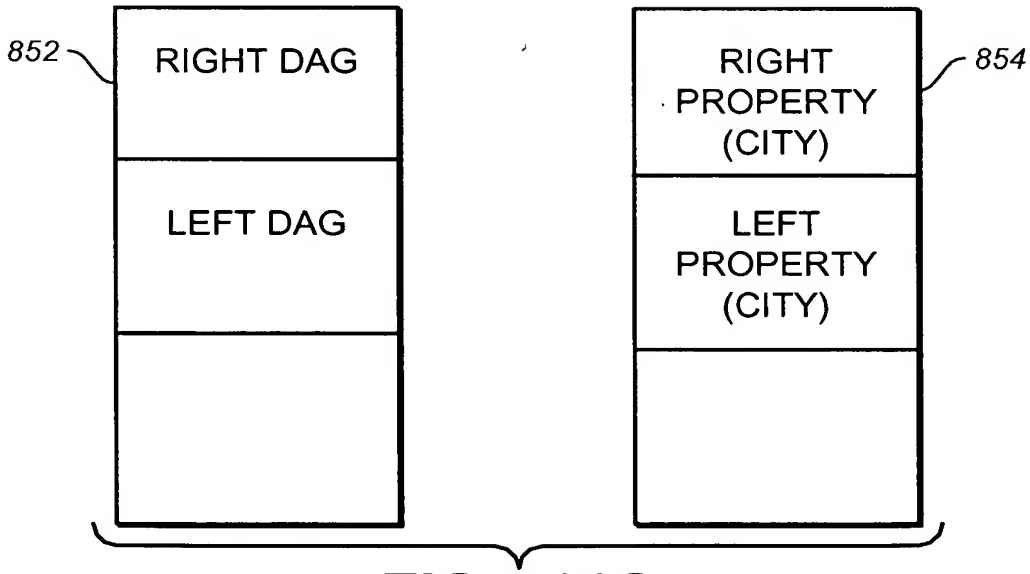


FIG._11G

FIG._13

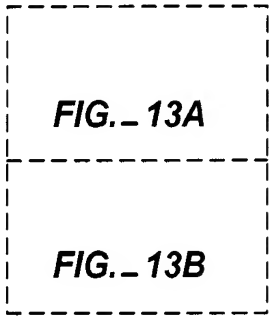
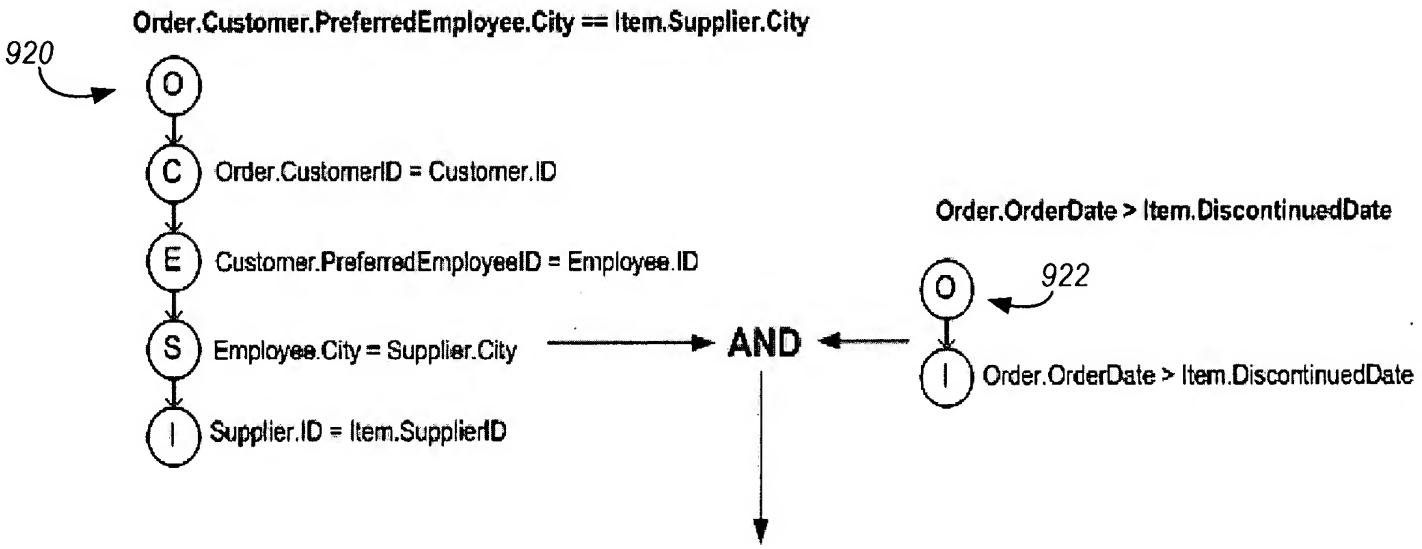


FIG._13A



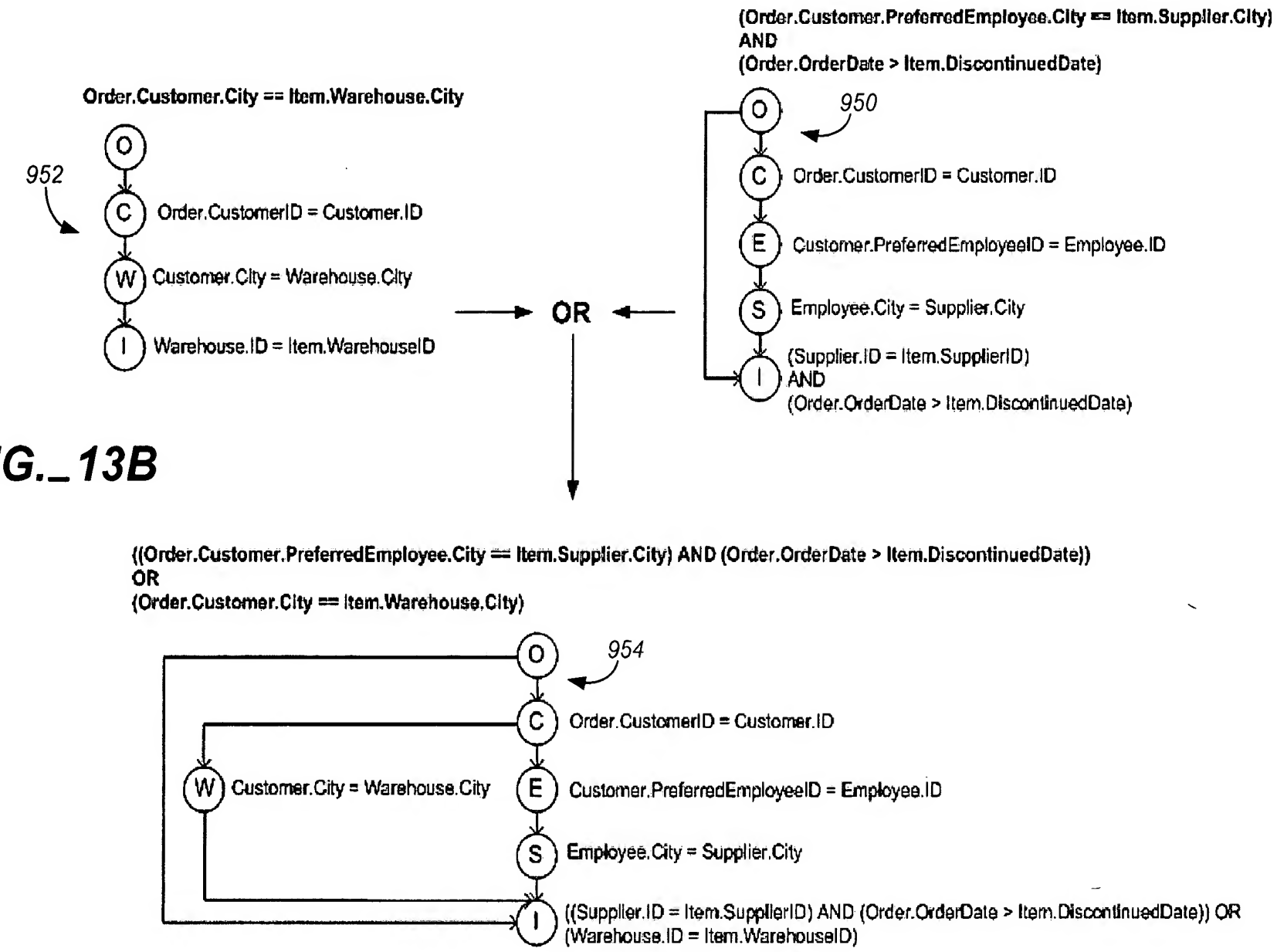

(Order.Customer.PreferredEmployee.City == Item.Supplier.City)
AND
(Order.OrderDate > Item.DiscontinuedDate)
950
O
C
Order.CustomerID = Customer.ID
E
Customer.PreferredEmployeeID = Employee.ID
S
Employee.City = Supplier.City
I
(Supplier.ID = Item.SupplierID)
AND
(Order.OrderDate > Item.DiscontinuedDate)
Order.Customer.City == Item.Warehouse.City
952
O
C
Order.CustomerID = Customer.ID
W
Customer.City = Warehouse.City
I
Warehouse.ID = Item.WarehouseID
OR
((Order.Customer.PreferredEmployee.City == Item.Supplier.City) AND (Order.OrderDate > Item.DiscontinuedDate))
OR
(Order.Customer.City == Item.Warehouse.City)
954
O
C
Order.CustomerID = Customer.ID
W
Customer.City = Warehouse.City
E
Customer.PreferredEmployeeID = Employee.ID
S
Employee.City = Supplier.City
I
((Supplier.ID = Item.SupplierID) AND (Order.OrderDate > Item.DiscontinuedDate)) OR
(Warehouse.ID = Item.WarehouseID)

FIG._13B

FIG._14

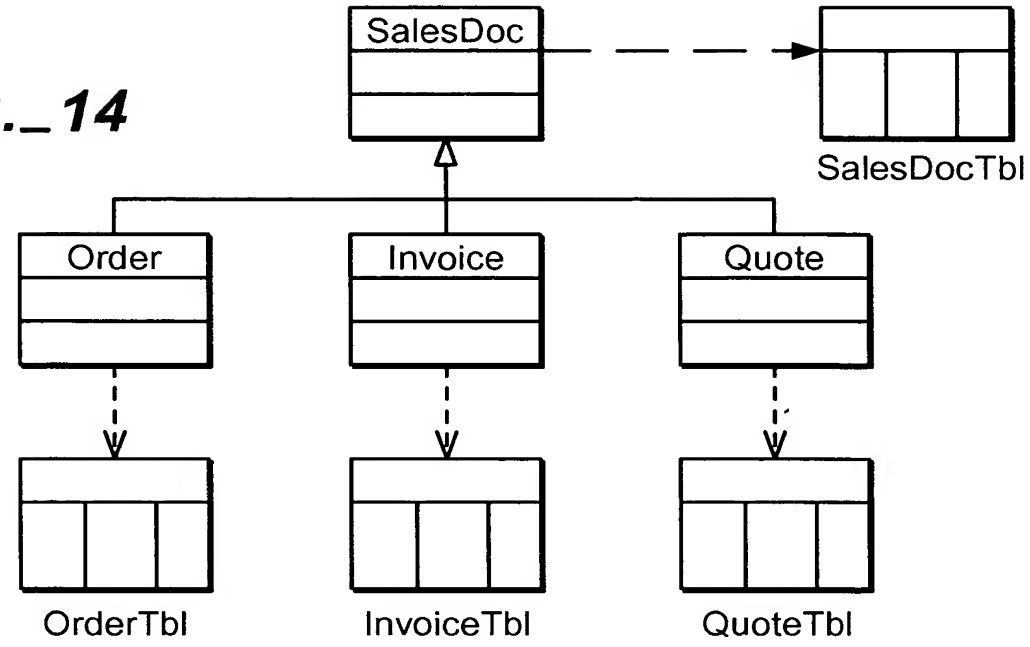


FIG._15

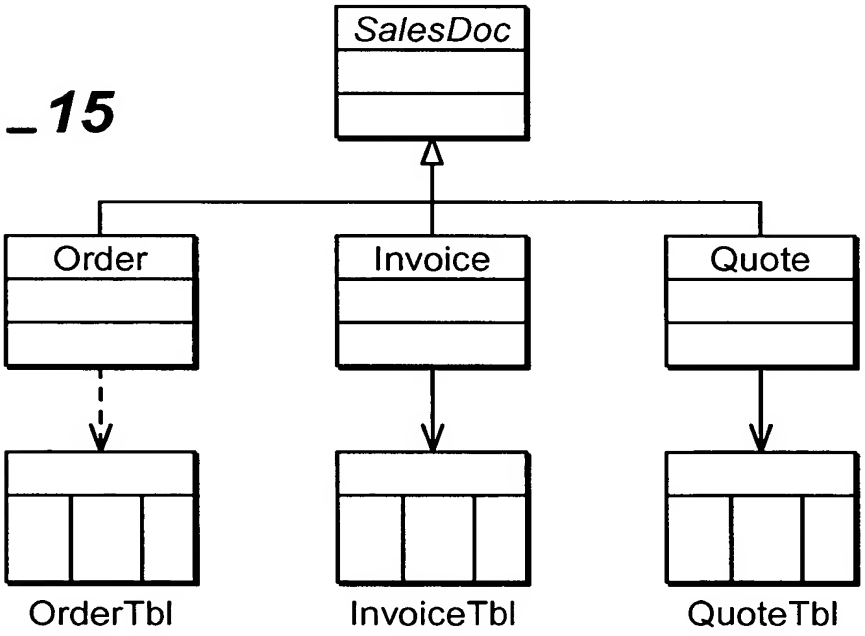
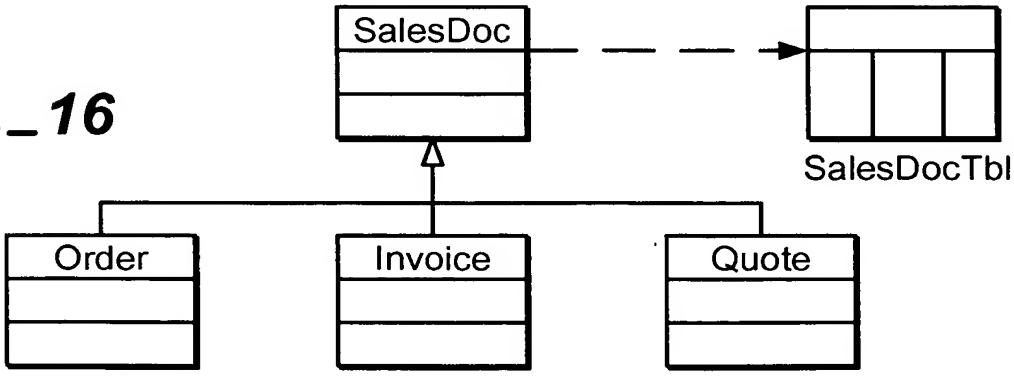


FIG._16



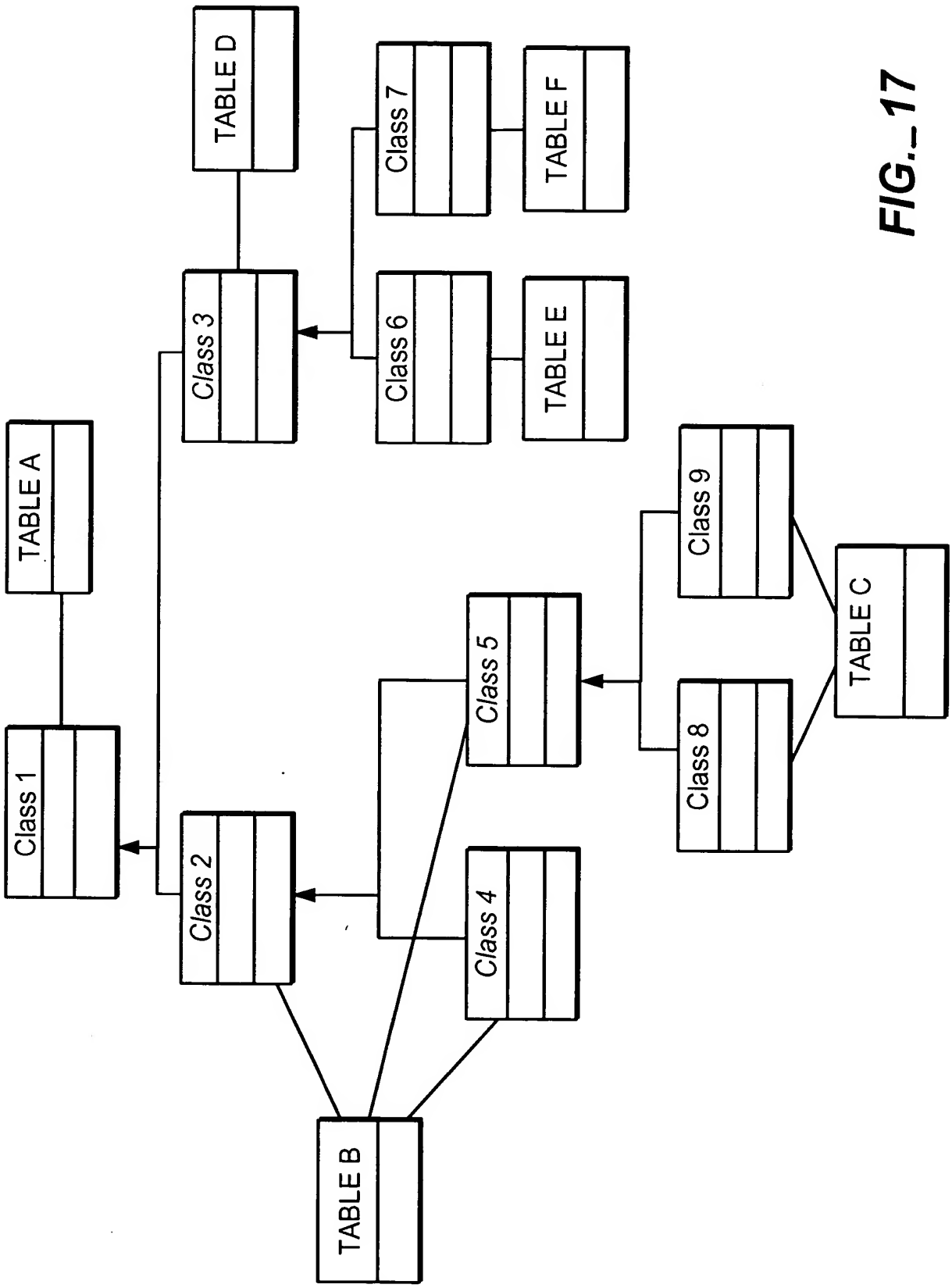


TABLE D
Class 7
TABLE F
Class 3
Class 6
TABLE E
TABLE A
Class 9
TABLE C
Class 5
Class 1
Class 8
Class 2
Class 4
TABLE B

FIG._17

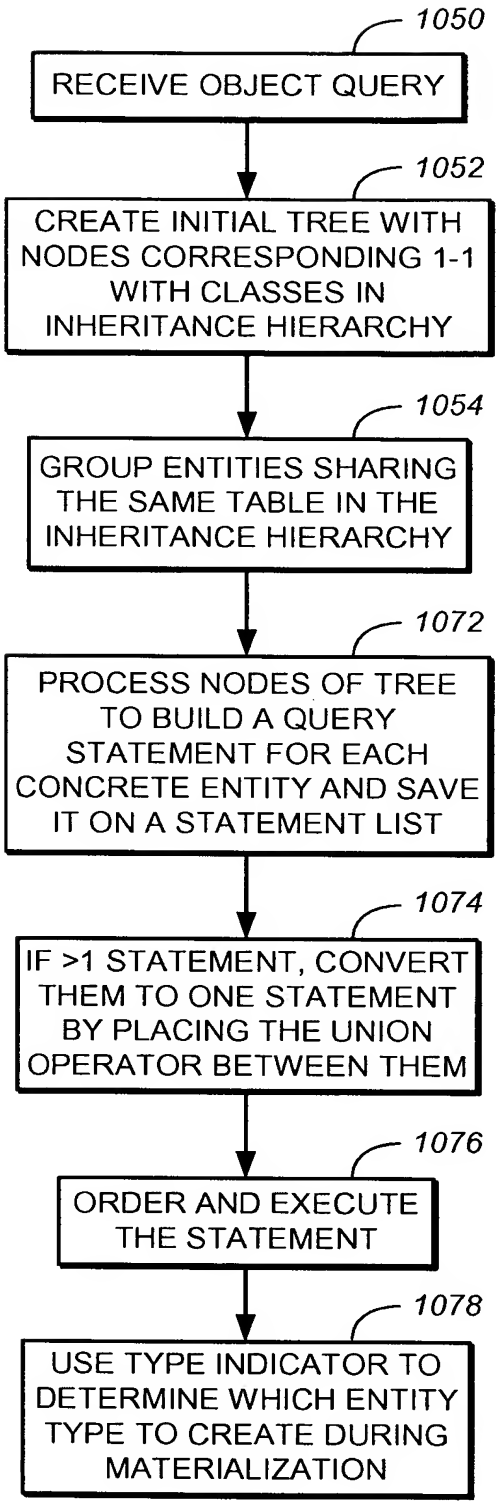


FIG._18

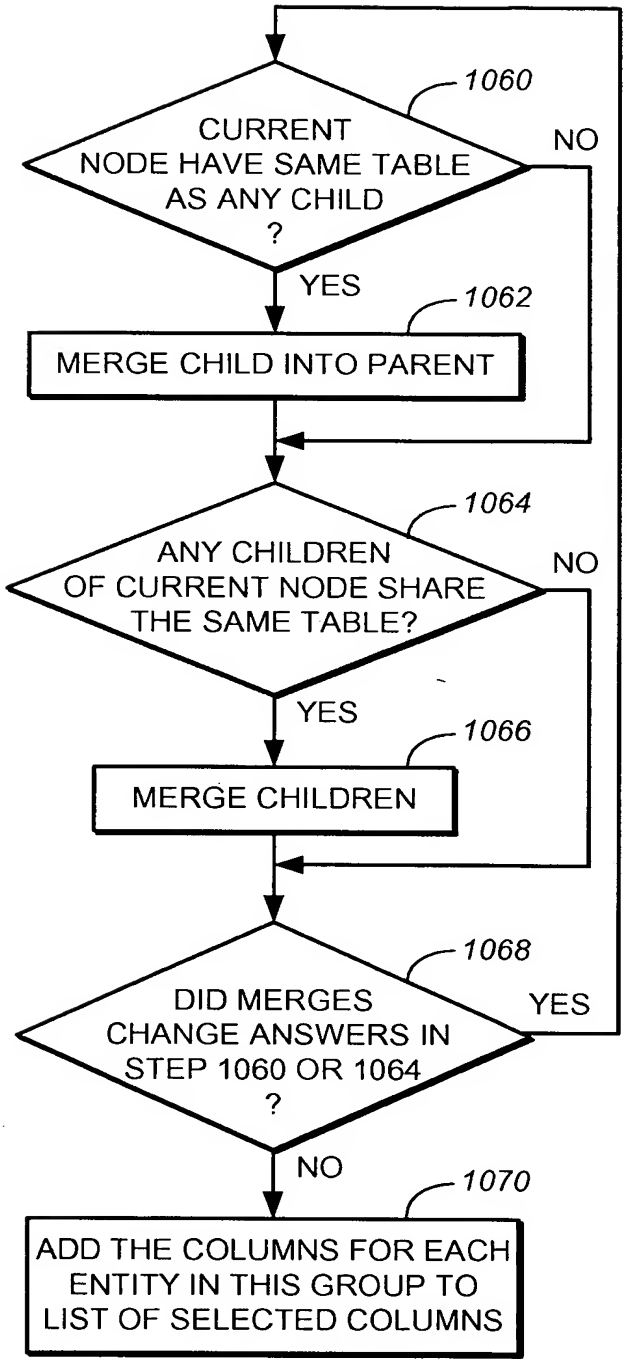


FIG._18-1

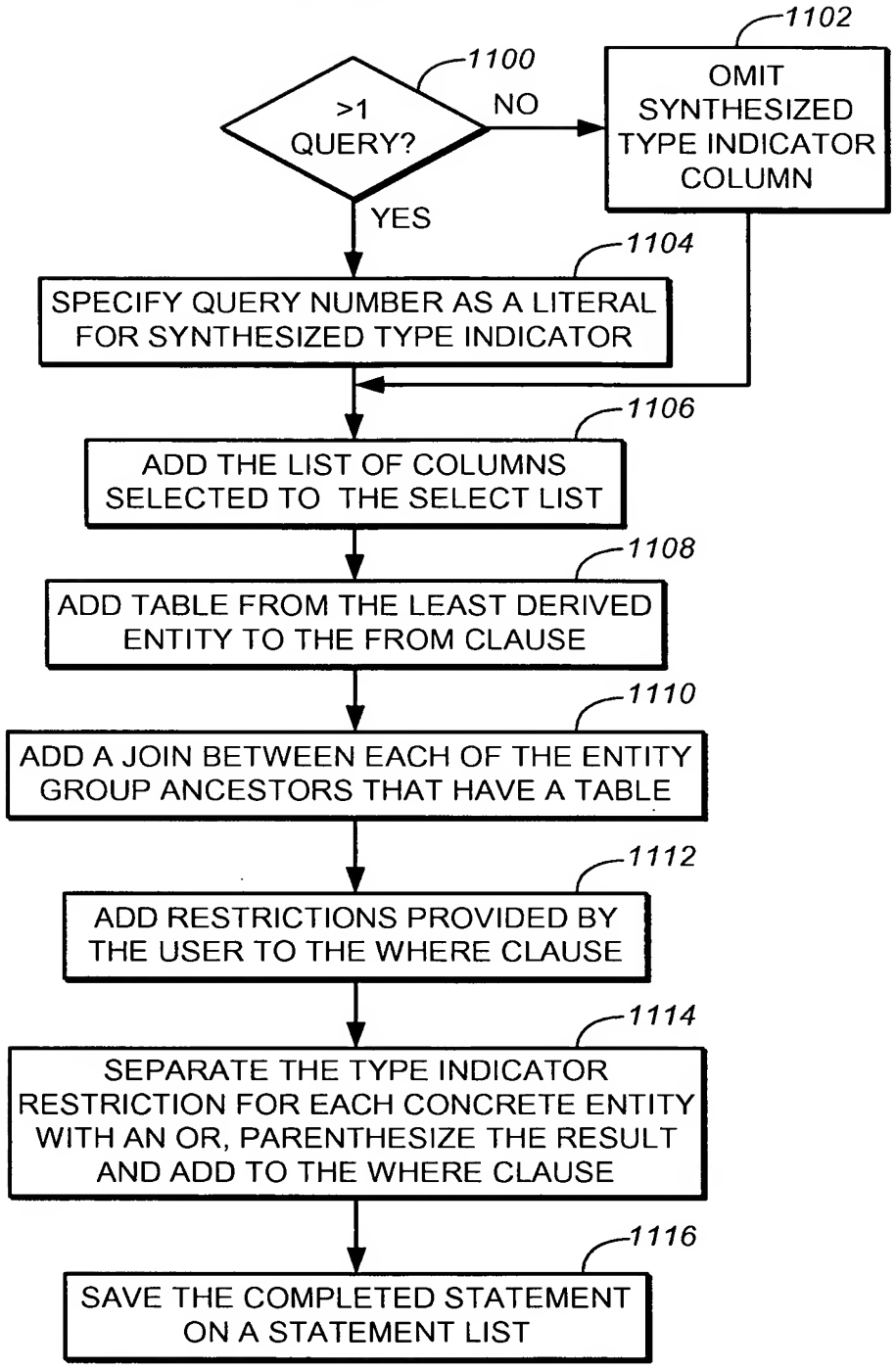

1100
>1 QUERY?
NO
1102
OMIT SYNTHESIZED TYPE INDICATOR COLUMN
YES
1104
SPECIFY QUERY NUMBER AS A LITERAL FOR SYNTHESIZED TYPE INDICATOR
1106
ADD THE LIST OF COLUMNS SELECTED TO THE SELECT LIST
1108
ADD TABLE FROM THE LEAST DERIVED ENTITY TO THE FROM CLAUSE
1110
ADD A JOIN BETWEEN EACH OF THE ENTITY GROUP ANCESTORS THAT HAVE A TABLE
1112
ADD RESTRICTIONS PROVIDED BY THE USER TO THE WHERE CLAUSE
1114
SEPARATE THE TYPE INDICATOR RESTRICTION FOR EACH CONCRETE ENTITY WITH AN OR, PARENTHESIZE THE RESULT AND ADD TO THE WHERE CLAUSE
1116
SAVE THE COMPLETED STATEMENT ON A STATEMENT LIST

FIG._18-2

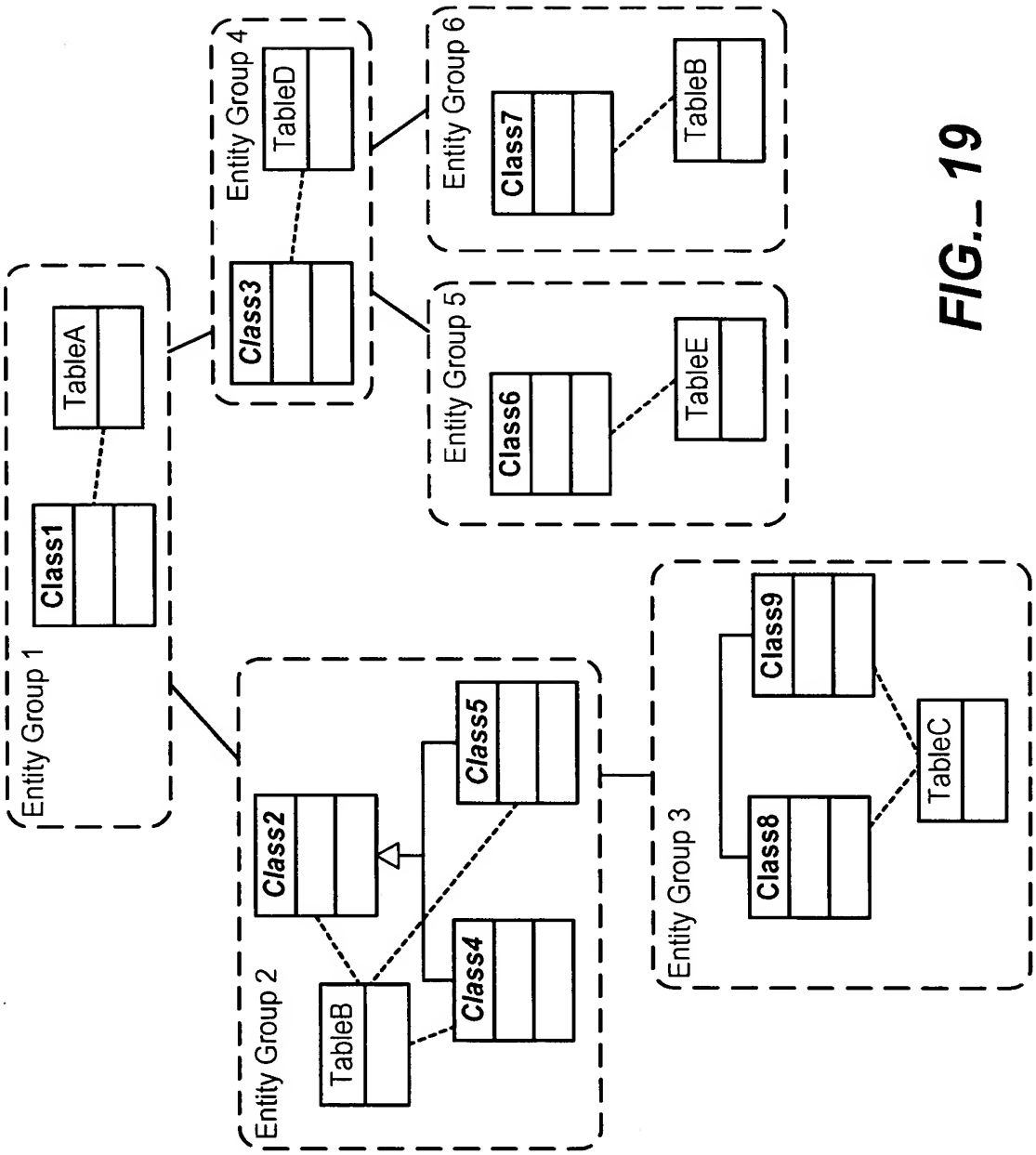

Entity Group 1
TableA
Class1
Entity Group 4
TableD
Class3
Entity Group 6
Class7
TableB
Entity Group 5
Class6
TableE
Entity Group 2
Class2
Class5
Class4
TableB
Entity Group 3
Class9
Class8
TableC

FIG._ 19

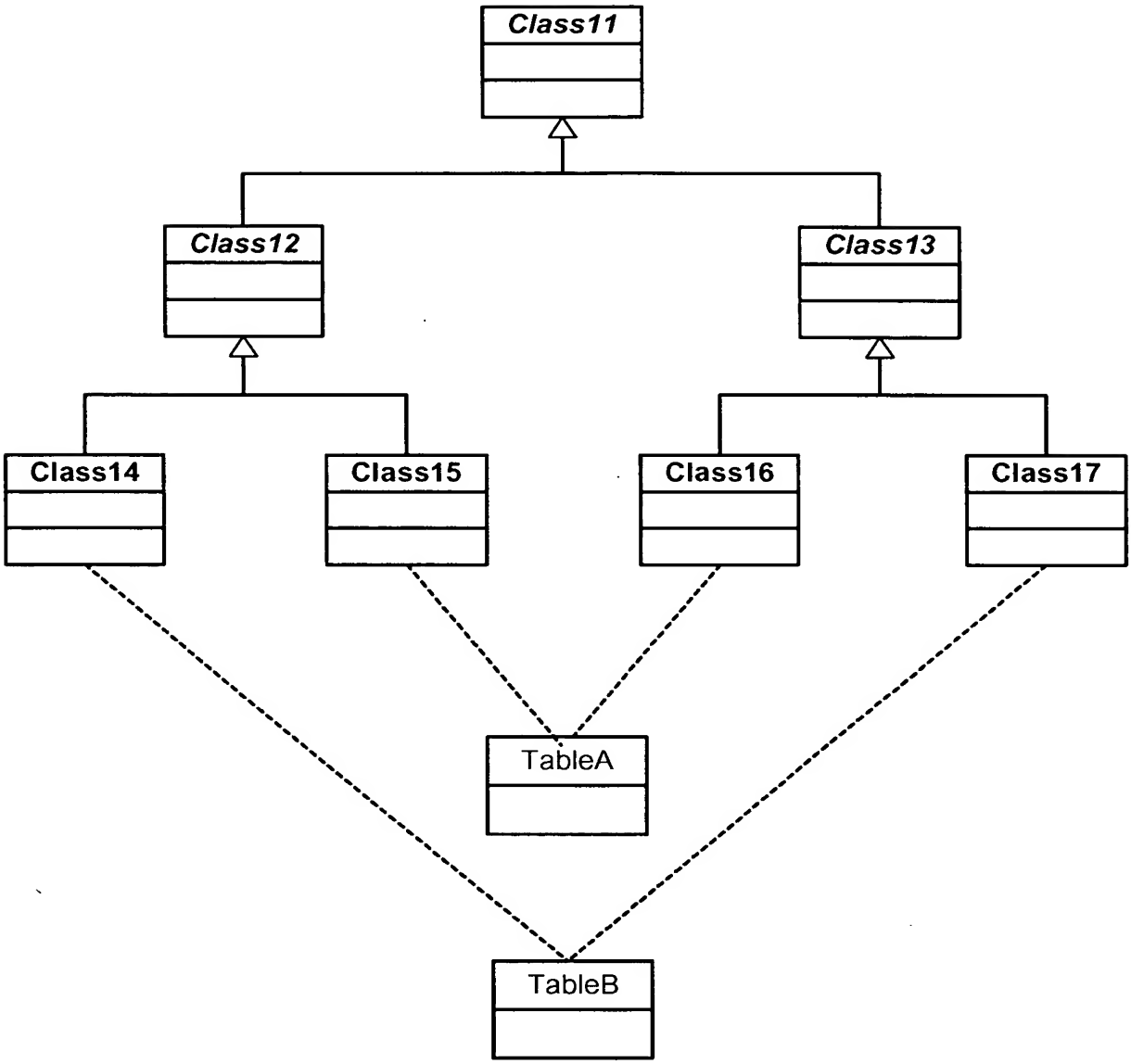

Class11
Class12
Class13
Class14
Class15
Class16
Class17
TableA
TableB

FIG._ 20

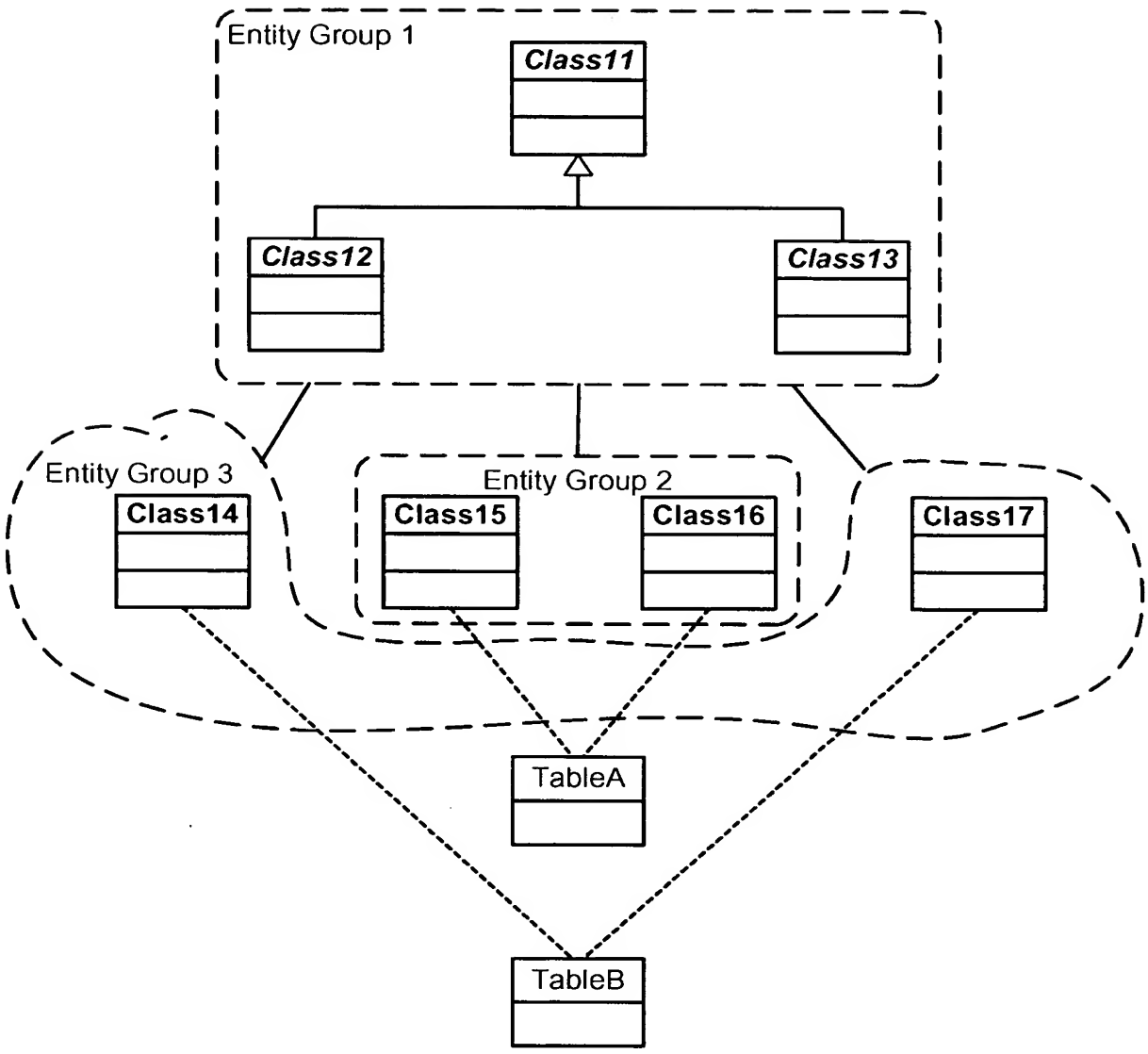

Entity Group 1
Class11
Class12
Class13
Entity Group 3
Class14
Entity Group 2
Class15
Class16
Class17
TableA
TableB

FIG._ 21

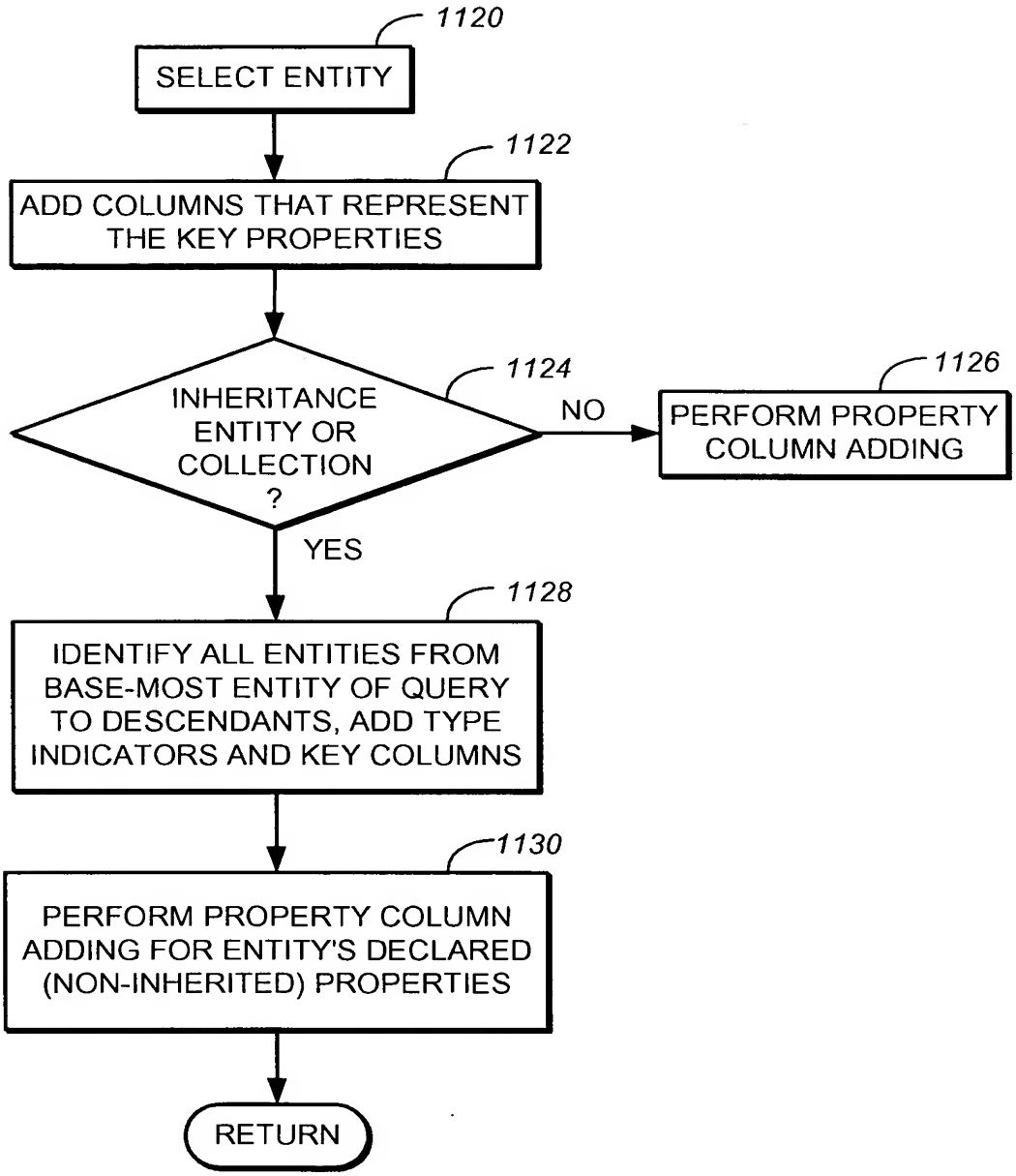

1120
SELECT ENTITY
1122
ADD COLUMNS THAT REPRESENT THE KEY PROPERTIES
1124
INHERITANCE ENTITY OR COLLECTION ?
NO
1126
PERFORM PROPERTY COLUMN ADDING
YES
1128
IDENTIFY ALL ENTITIES FROM BASE-MOST ENTITY OF QUERY TO DESCENDANTS, ADD TYPE INDICATORS AND KEY COLUMNS
1130
PERFORM PROPERTY COLUMN ADDING FOR ENTITY'S DECLARED (NON-INHERITED) PROPERTIES
RETURN

FIG._ 22

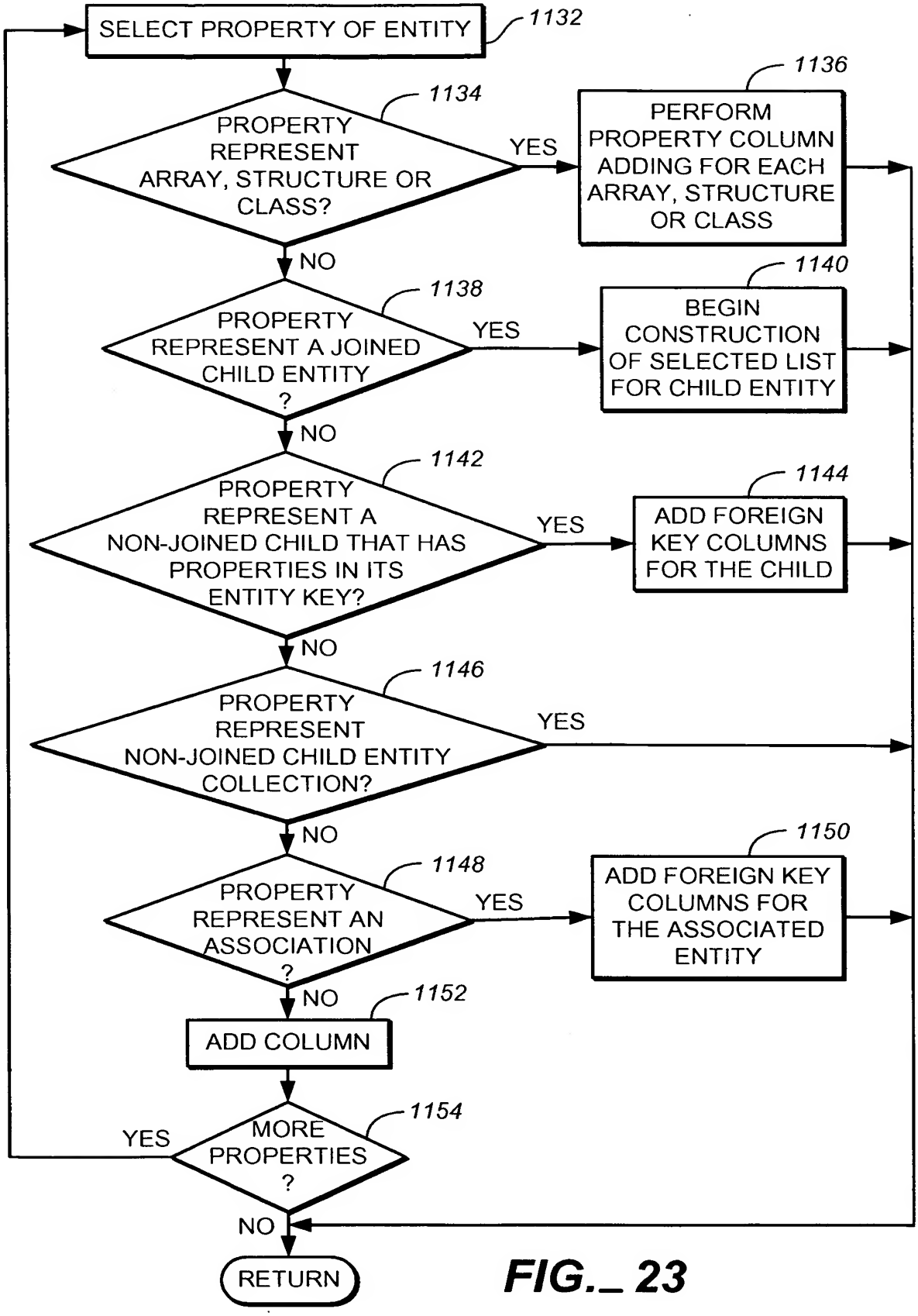

SELECT PROPERTY OF ENTITY
1132
PROPERTY REPRESENT ARRAY, STRUCTURE OR CLASS?
1134
YES
PERFORM PROPERTY COLUMN ADDING FOR EACH ARRAY, STRUCTURE OR CLASS
1136
NO
PROPERTY REPRESENT A JOINED CHILD ENTITY ?
1138
YES
BEGIN CONSTRUCTION OF SELECTED LIST FOR CHILD ENTITY
1140
NO
PROPERTY REPRESENT A NON-JOINED CHILD THAT HAS PROPERTIES IN ITS ENTITY KEY?
1142
YES
ADD FOREIGN KEY COLUMNS FOR THE CHILD
1144
NO
PROPERTY REPRESENT NON-JOINED CHILD ENTITY COLLECTION?
1146
YES
NO
PROPERTY REPRESENT AN ASSOCIATION ?
1148
YES
ADD FOREIGN KEY COLUMNS FOR THE ASSOCIATED ENTITY
1150
NO
ADD COLUMN
1152
MORE PROPERTIES ?
1154
YES
NO
RETURN

FIG._ 23

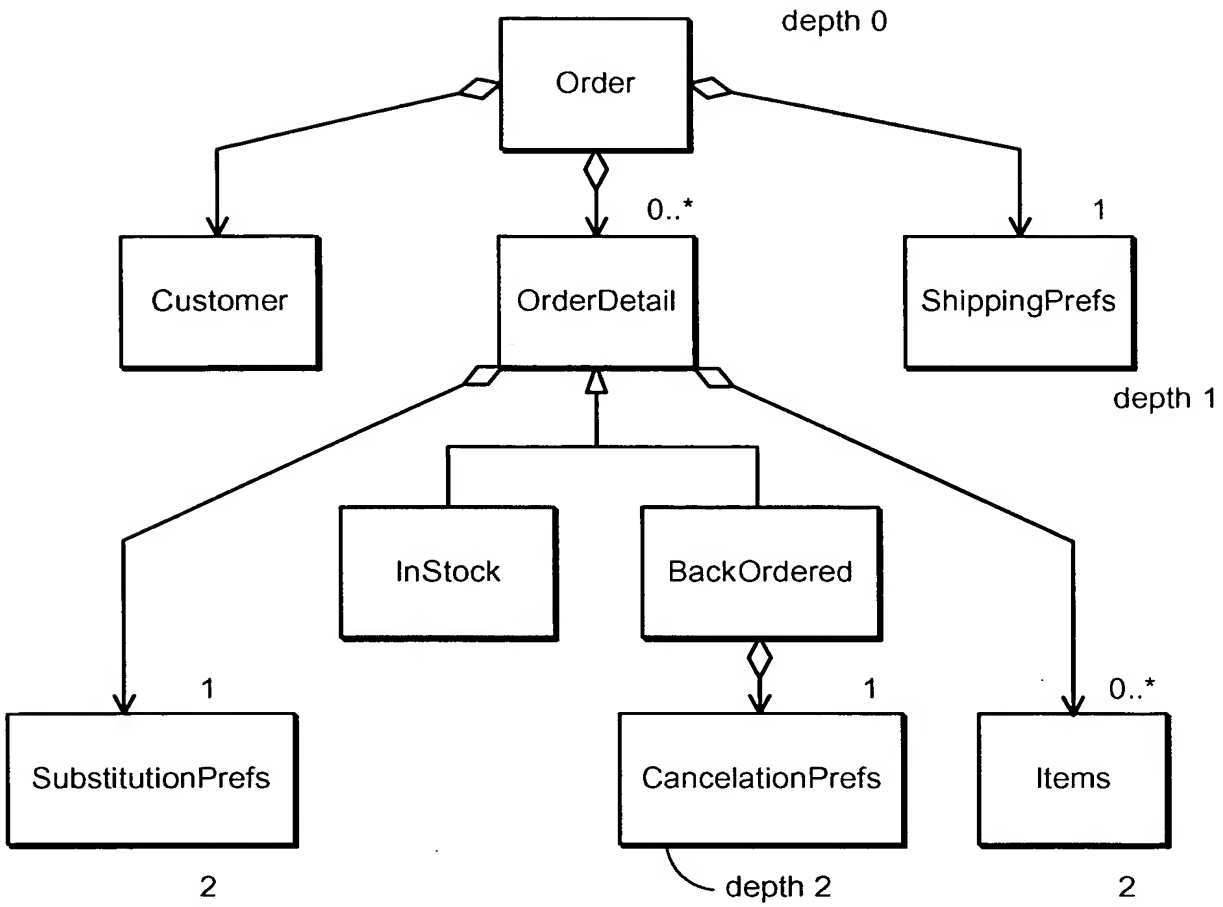

depth 0
Order
0..*
1
Customer
OrderDetail
ShippingPrefs
depth 1
InStock
BackOrdered
1
1
0..*
SubstitutionPrefs
CancelationPrefs
Items
depth 2
2
2

FIG._ 24

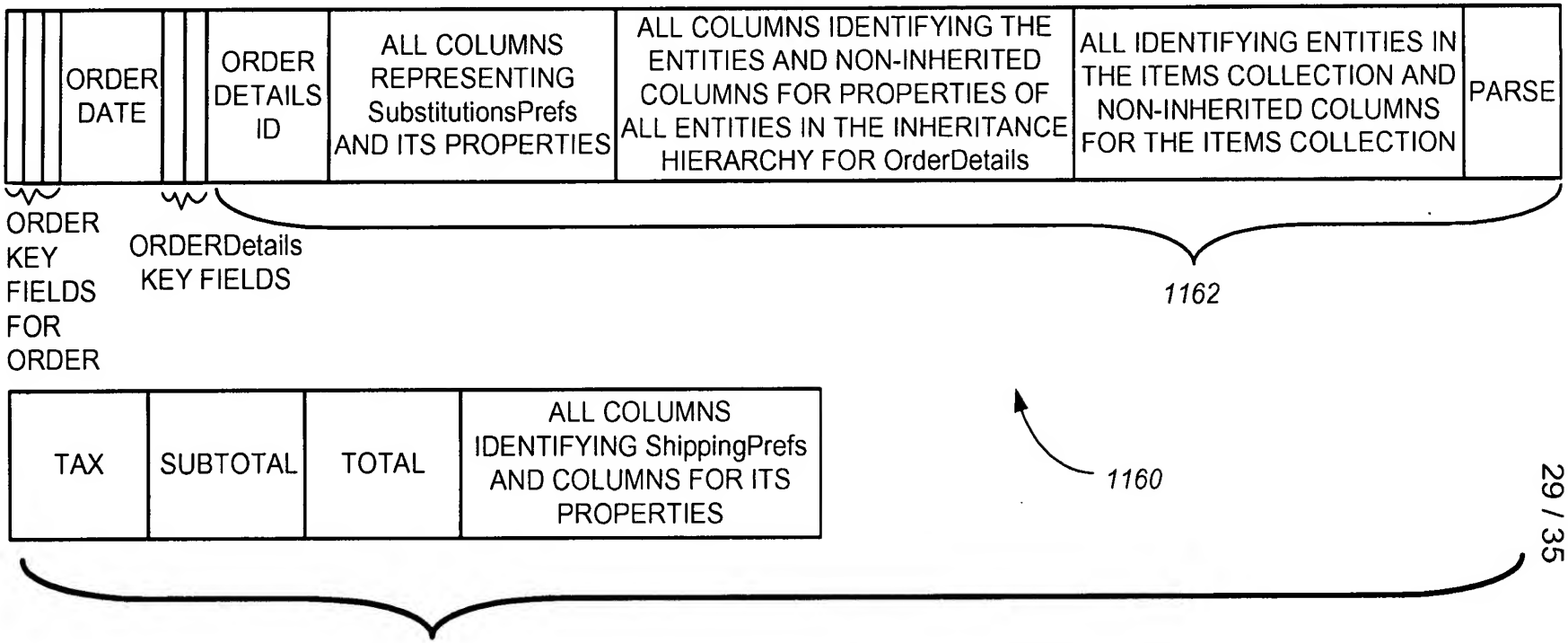


FIG._ 25

```
CLASS ORDER {
        ID
        DATE
        OrderDDetails
        Collection
        Tax
        Subtotal
        Total
        ShippingPrefs
}
```

FIG._25A

```
CLASS OrderDetail {
        ID
        SubstitutionPrefs
        Items
        Collection
        Misc
}
```

FIG._25B

212

```
1002  class Order {            // these fields are mapped to a database table public string ID;
          ...
          public DetailCollection Details;   // contains Detail objectspublic decimal Total;}
      class Detail {       // these fields are mapped to a database table
1004      public long SequenceNumber;
          public Item Item;
          public decimal PricePerUnit;
          public decimal Quantity;
1006      public decimal Price;                          // PricePerUnit * Quantity}

      // the user describes the set operation they want performed in terms of objects
      Criteria.EntitySetUpdateCriteria(Criteria.EntityAlias(parentKey, typeof(Order))),
1010  // update the order// set Order.Total to the sum of each of the line item's price
          Criteria.PropertyAssignments(Criteria.Assignment((Property)"Order.Total",
      Criteria.Sum((Property)"Order.Details[].Price")       )
1008  // indicates which orders to update; only those with detail price > 300
          Criteria.Where((Property)"Order.OrderDetails[].Price" > 300),)
```

FIG._ 26

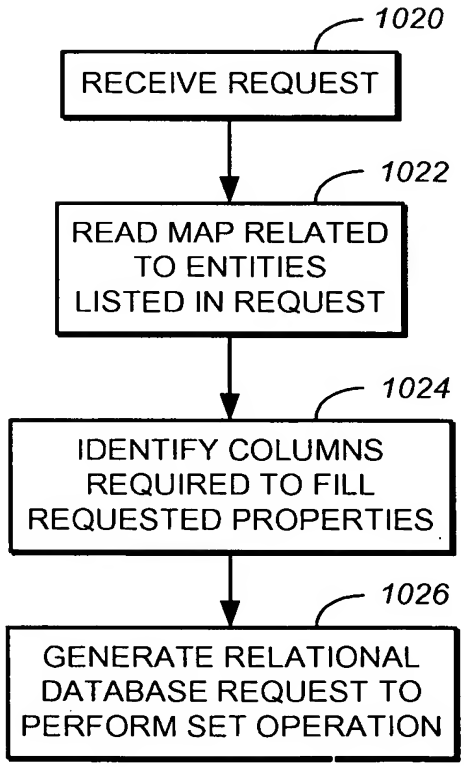


FIG._ 27

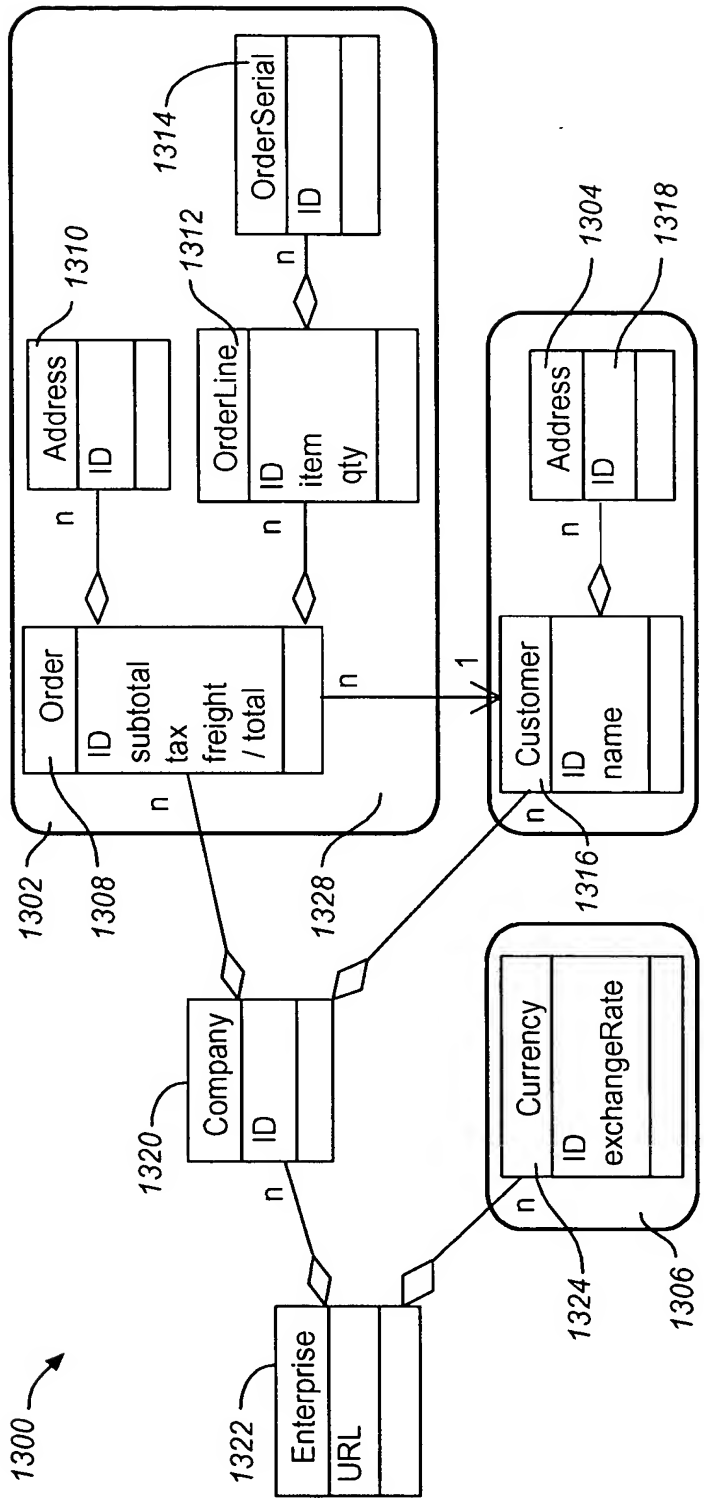

1300
1302
1304
1306
1308
1310
1312
1314
1316
1318
1320
1322
1324
1328
Order
ID
subtotal
tax
freight
/ total
Address
ID
OrderLine
ID
item
qty
OrderSerial
ID
Customer
ID
name
Address
ID
Company
ID
Currency
ID
exchangeRate
Enterprise
URL
n
1

FIG._ 28

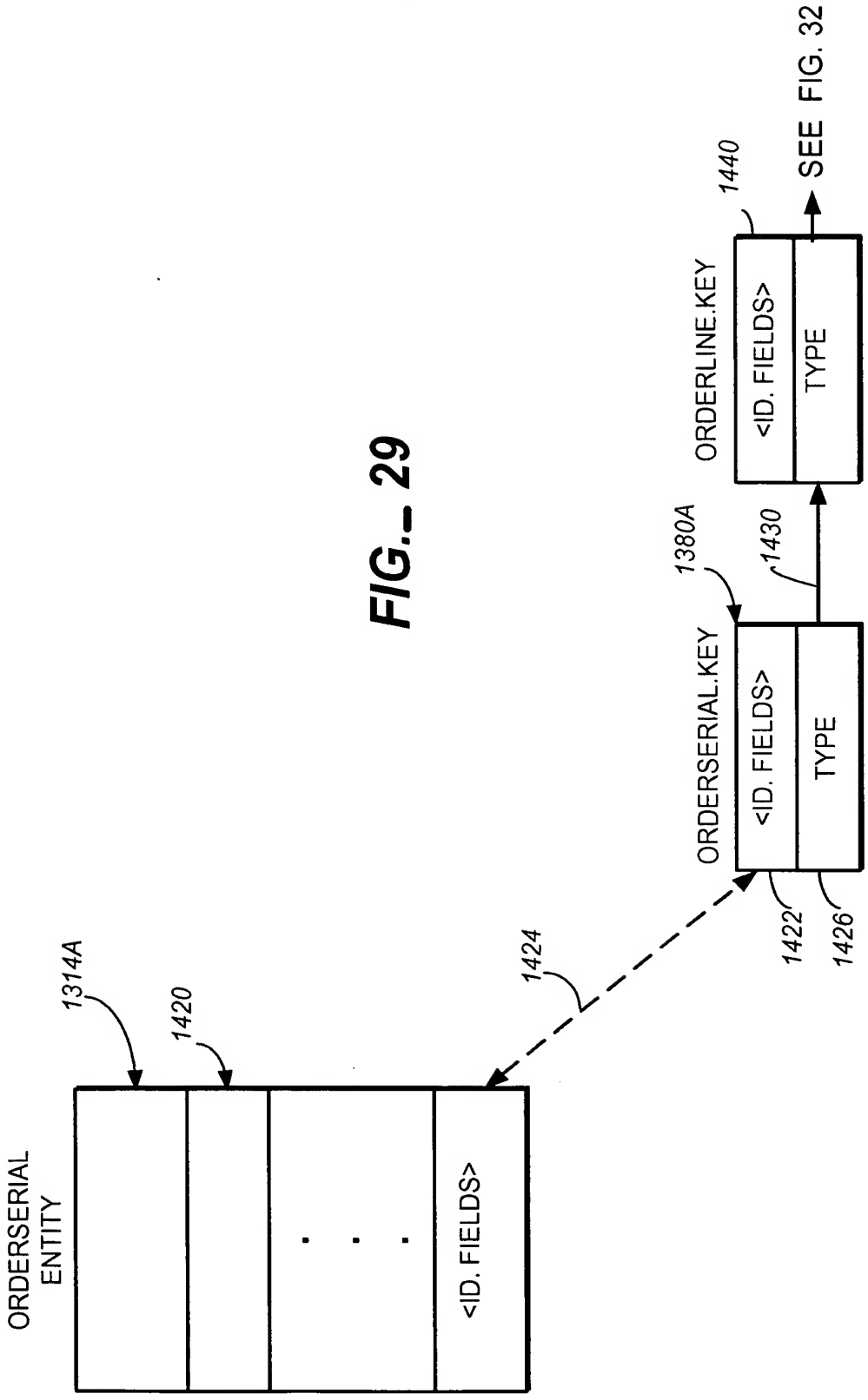

ORDERSERIAL ENTITY
<ID. FIELDS>
1314A
1420
1424
ORDERSERIAL.KEY
<ID. FIELDS>
TYPE
1380A
1430
1422
1426
ORDERLINE.KEY
<ID. FIELDS>
TYPE
1440
SEE FIG. 32

FIG._29

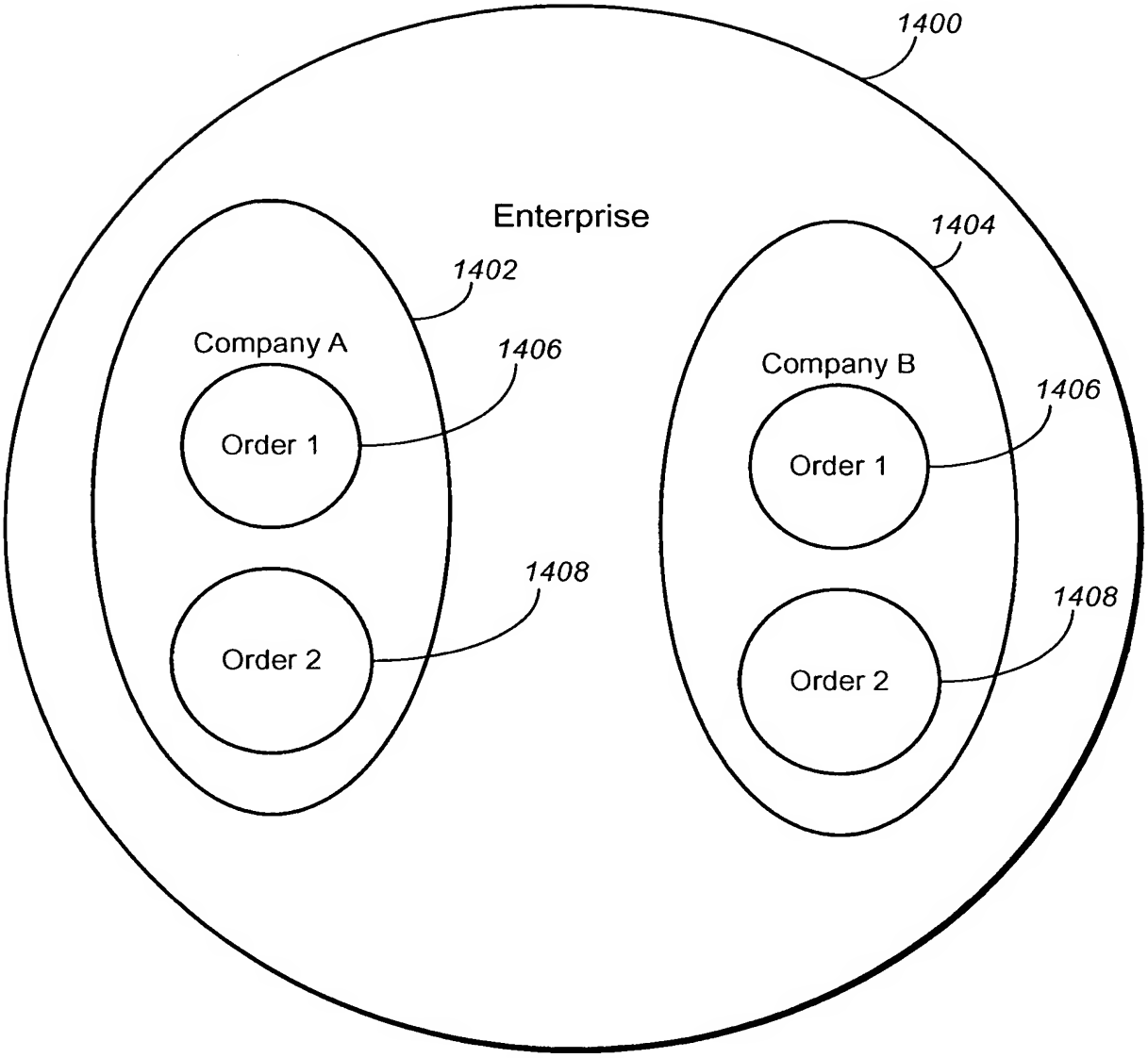

1400
Enterprise
1402
Company A
1406
Order 1
1408
Order 2
1404
Company B
1406
Order 1
1408
Order 2

FIG._ 30

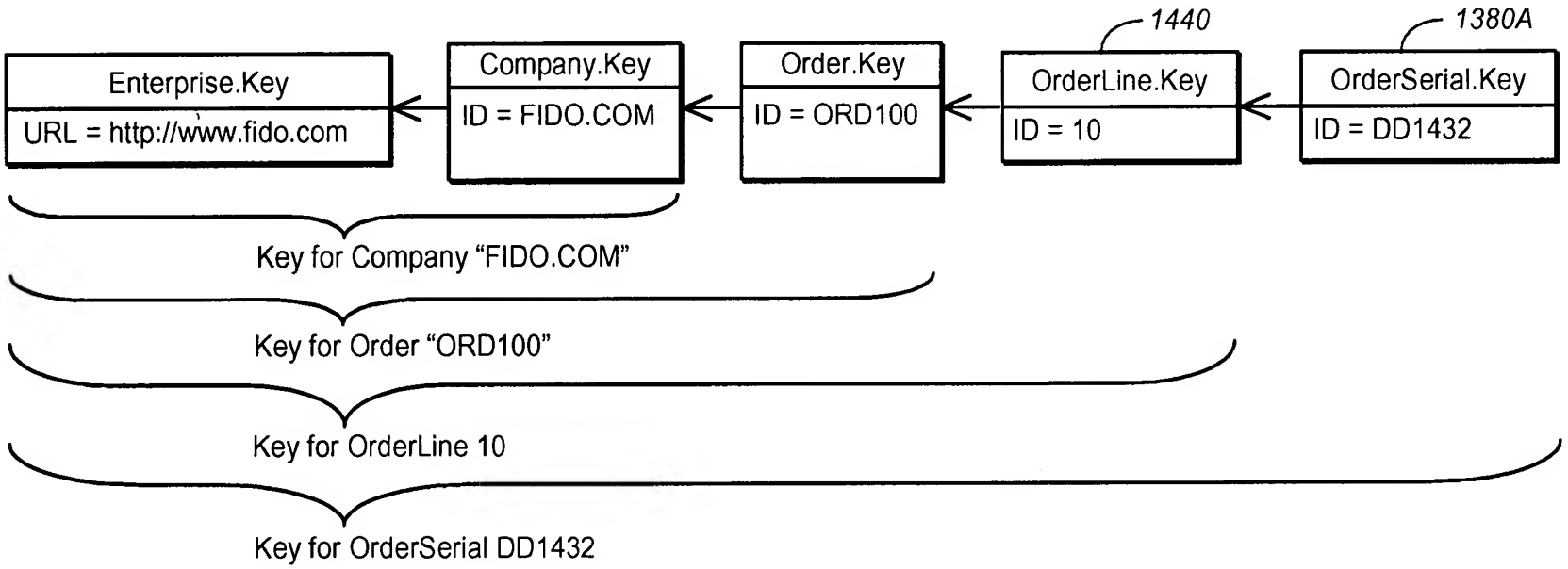

1440
1380A
Enterprise.Key
URL = http://www.fido.com
Company.Key
ID = FIDO.COM
Order.Key
ID = ORD100
OrderLine.Key
ID = 10
OrderSerial.Key
ID = DD1432
Key for Company "FIDO.COM"
Key for Order "ORD100"
Key for OrderLine 10
Key for OrderSerial DD1432

FIG._ 31

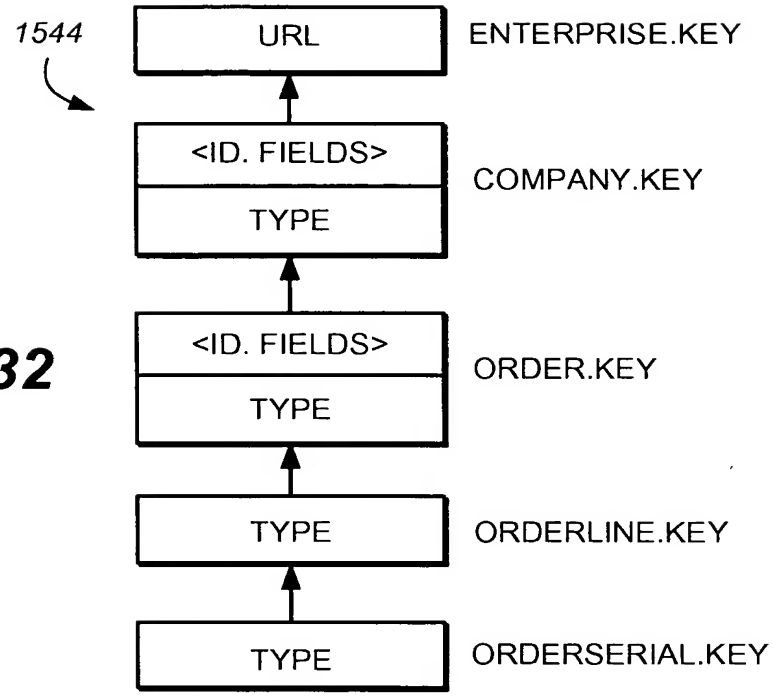


FIG._ 32

1350

COMPANY_ID 1352	ORDER_ID 1354	ORDERLINE_ID 1356	SERIAL NO. 1358	OTHER COLUMNS	
⋮	⋮	⋮	⋮	⋮	⋮

FIG._ 33